

# RECONOCIMIENTO DE IMÁGENES UTILIZANDO REDES NEURONALES ARTIFICIALES

PEDRO PABLO GARCÍA GARCÍA

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA, FACULTAD DE INFORMÁTICA,  
UNIVERSIDAD COMPLUTENSE DE MADRID



Proyecto Fin de Máster en Ingeniería Informática para la Industria  
“Máster en Investigación en Informática, Facultad de Informática, Universidad  
Complutense de Madrid”

Autor: **Pedro Pablo García García**

Tutor: **José Antonio López Orozco**

Curso 2012/2013



## **Autorización de Difusión**

PEDRO PABLO GARCÍA GARCÍA

2012/2013

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: “RECONOCIMIENTO DE IMÁGENES UTILIZANDO REDES NEURONALES ARTIFICIALES”, realizado durante el curso académico 2012-2013 bajo la dirección de José Antonio López Orozco en el Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.



## **Resumen**

Este trabajo describe el proceso de extracción de patrones característicos de imágenes, mediante la ayuda de Redes Neuronales Artificiales. La información de la Red Neuronal junto con datos adicionales de las imágenes, serán almacenados en una base de datos y consumidos por un servicio web. Un teléfono móvil con sistema operativo Android consumirá la información almacenada en el servicio web. Posteriormente al realizar una captura de imagen con la cámara del teléfono, este procesará la imagen y junto con los datos consumidos por el servicio web será capaz identificar de qué imagen se trata. Para el tratamiento de las imágenes se utilizarán librerías OpenCV, tanto en el servidor como en el teléfono móvil.

## **Palabras clave**

OpenCV; Hu moments; Soap; Red Neuronal Backpropagation.

## **Summary**

This essay describes the process of taking out patterns included in images by using Artificial Neural Networks. Data contained in these networks along with information from images will be stored in a database and used by a web service. Any Android-operated cell phone will be able to use the web service's data when a picture is taken in the unit. Images taken with the cell phone's built-in camera will be processed using the data contained in the neural network. As a result, the cell phone will be able to identify what is in the picture. Image processing will use Open CV libraries from both the cell phone and the web service.

## **Keywords**

OpenCV; Hu moments; Soap; Red Neuronal Backpropagation.

## **Agradecimientos**

He de expresar mi profundo agradecimiento a todas aquellas personas que me han ayudado a llevar a cabo mi trabajo, sin ellas no hubiese sido posible.

José Antonio López Orozco mi director de proyecto, cuya importancia ha sido vital en el desarrollo del mismo. Me ha aportado sus conocimientos y experiencia y gracias a él he conseguido iniciarme en el mundo de la investigación. Su influencia ha sido de vital importancia para la realización de este trabajo.

Gonzalo Pajares, experto en tratamiento digital de imágenes que me aportó valiosos consejos para la consecución del proyecto.

Diego Martínez Plasencia doctor en realidad virtual, quien también me aportó valiosos consejos para este proyecto.

Por último, en el aspecto personal, mi gratitud a mi familia por apoyarme en todo momento para seguir adelante.





## Índice de contenidos

<b>AUTORIZACIÓN DE DIFUSIÓN .....</b>	<b>3</b>
<b>RESUMEN.....</b>	<b>5</b>
<b>PALABRAS CLAVE.....</b>	<b>5</b>
<b>SUMARY.....</b>	<b>6</b>
<b>KEYWORDS .....</b>	<b>6</b>
<b>AGRADECIMIENTOS .....</b>	<b>7</b>
<b>ÍNDICE DE CONTENIDOS.....</b>	<b>9</b>
1. INTRODUCCIÓN .....	11
1.1 Motivación .....	11
1.2 Objetivos .....	11
1.3 Estado del arte .....	12
1.4 Estructura del trabajo .....	19
2 DISEÑO DEL PROYECTO .....	21
2.1 Infraestructura del proyecto .....	21
2.2 Herramientas y tecnologías utilizadas .....	23
3 TRATAMIENTO DE IMAGEN.....	27
3.1 Escala de Grises.....	27
3.2 Filtrado de imagen .....	28
3.3 Binarización.....	29
3.4 Detección de bordes (Canny).....	30
3.5 Detección de contorno .....	31
3.6 Extracción de características.....	32
3.7 Obtención red neuronal .....	35
4 EVALUACIÓN DE RESULTADOS.....	39
4.1 Estudio de la red neuronal.....	39
4.2 Estudio de aplicación Android .....	49
5 CONCLUSIONES Y TRABAJO FUTURO .....	51
<b>BIBLIOGRAFÍA.....</b>	<b>53</b>

[ANEXO A.1] .....	55
[ANEXO A.2] .....	57
[ANEXO A.3] .....	63
[ANEXO A.4] .....	71

## **1. Introducción**

El presente capítulo describe el contexto del trabajo desarrollado seguido de los objetivos del mismo y finaliza con su estructura.

### **1.1 Motivación**

Muchas veces hemos ido por la calle hemos visto un objeto que no reconocemos y no lo hemos podido buscar porque no sabemos bajo qué nombre buscarlo, o hemos ido por la calle y hemos visto un monumento y no sabíamos qué era, o hemos estado en un museo y nos hubiera gustado tener más información al instante sobre la obra que contemplamos. De todo esto los gigantes de la informática se han dado cuenta y han lanzando aplicaciones de reconocimiento de imágenes para dispositivos móviles desarrollando algoritmos basados en el contenido de las imágenes reconociendo formas y analizando su parecido. Sin necesidad de conocer el nombre de una pieza, sin tener que teclear o escribir datos, sólo haciendo una foto con nuestro teléfono podremos obtener toda la información sobre ese objeto. Sin embargo, se echa de menos aplicaciones genéricas de reconocimiento de imágenes en las que el servidor sea capaz de reconocer imágenes utilizando mínimos recursos.

El trabajo de investigación desarrollado tiene su origen en la necesidad de desarrollar aplicaciones para teléfonos móviles que sean capaces de identificar imágenes de una forma rápida y sin consumo excesivo de recursos, tanto de comunicación con un servidor remoto como de recursos del propio móvil.

### **1.2 Objetivos**

Los objetivos planteados en este trabajo de investigación son los siguientes:

1. Desarrollar un sistema de reconocimiento de imágenes para dispositivos móviles, el cual debe ser capaz de funcionar en modo local, sin una conexión permanente con el servidor, además debería poder adaptarse a cualquier tipo de imagen que se quiera identificar. El sistema debe ser robusto y versátil.

2. Crear una infraestructura adecuada para la creación de un sistema de identificación de imágenes para teléfonos móviles que sea genérico, de forma que se pueda utilizar para el reconocimiento de cualquier tipo de imagen.

### 1.3 Estado del arte

Los buscadores de imágenes como *Google images*, se basan principalmente en el nombre del archivo que contiene la imagen, en el nombre del enlace o en el texto que aparece en la página donde se encuentra la imagen. Es decir que la búsqueda se basa en información de texto y no en información gráfica.

Actualmente se han desarrollado aplicaciones como *Google Goggles*<sup>1</sup> basadas en el reconocimiento de imágenes utilizando el contenido de las mismas, para ello se centran en un subconjunto de las imágenes que Google ha catalogado previamente ya que el análisis y comparación de imágenes digitales requiere de un coste computacional muy alto. Realizar esta labor para todas las imágenes indexadas por un buscador sería una tarea casi imposible, a día de hoy.

El campo de procesamiento de imágenes está continuamente evolucionando. Durante los últimos años ha habido un incremento significativo en el interés en campos como morfología de imágenes, redes neuronales artificiales [\[ANEXO A.1\]](#), procesamiento de imágenes en color y/o en escala de grises, comprensión de datos de imágenes, reconocimiento de imágenes y sistemas de análisis basados en conocimiento.

Se puede encontrar una gran cantidad de aplicaciones de reconocimiento de imágenes en el mercado, por destacar algunas, podríamos citar:

---

<sup>1</sup> Servicio de Google que permite el reconocimiento de cualquier objeto mediante fotos realizadas con un móvil

- Reconocimiento de rostros y expresiones faciales.
- Reconocimiento de firmas.
- Reconocimiento de caracteres.

### 1.3.1 Reconocimiento de rostros y expresiones faciales

Para los humanos, la forma más sencilla de reconocer personas es a través de su rostro, ya que tiene características únicas como distancia entre ojos, anchura de la nariz, forma de la barbilla, pómulos, forma de la boca, etc.

Se pueden encontrar numerosos trabajos relacionados con el reconocimiento facial [\[1\]](#) [\[2\]](#) [\[3\]](#).

En este apartado se describe brevemente cómo se ha desarrollado un sistema computacional de seguridad basado en el reconocimiento de rostros [\[1\]](#). El sistema estudiado está conformado por una computadora conectada a una cámara web colocada sobre un soporte, el cual a su vez tendrá una base donde se coloca el rostro para que la cámara pueda capturar la imagen, una vez capturada la imagen, ésta es enviada a la computadora para que proceda al estudio de la imagen.

El programa que estudia la imagen, permite segmentar las imágenes capturadas para obtener únicamente las imágenes del rostro y poder así descartar información no relevante de la imagen, posteriormente se procede a la descomposición de las imágenes para luego aplicarles la técnica de Análisis de Componentes Principales, para finalmente reconocer la imagen.

La segmentación de las imágenes está basada en las proyecciones de las derivadas de las filas y columnas de los valores de la imagen.

Una vez la imagen ha sido segmentada se procede a descomponer la imagen en cuatro sub-imágenes que contienen detalles principales del rostro.

Por último se procede a realizar el Análisis de Componentes Principales que consiste en la recolección de imágenes de rostros de varias personas que son luego combinadas y convertidas en una matriz. Los vectores que conforman la matriz pueden ser combinados adecuadamente para reconstruir adecuadamente cualquier imagen facial del conjunto.

### 1.3.2 Reconocimiento de firmas

Actualmente el reconocimiento de firmas y la seguridad se ha convertido en un tema prioritario para garantizar la protección de los sistemas frente a una mala utilización voluntaria, por lo que resulta imprescindible realizar un proceso de identificación y verificación de la identidad de cada persona de modo seguro, pero a la vez sencillo y natural.

**La Biometría** facilita la identificación de cada individuo de forma unívoca mediante la medición de diferentes características personales e intransferibles. Estas características individuales se clasifican en **características físicas** (huella dactilar, iris, retina, etc.); y **características del comportamiento** (forma de hablar, escribir, firmar, teclear, etc.)

Existen gran cantidad de trabajos relacionados en esta área [\[4\]](#)[\[5\]](#)

En [\[4\]](#) se utilizan sistemas de identificación Biométricos<sup>2</sup> para la identificación de las firmas de cada individuo. Para ello se parte de una base de datos biométricos multimodal inicial en las que había 3750 firmas correspondientes a 75 individuos. La mitad de ellas eran firmas verdaderas y la otra mitad falsas. En este artículo se discuten dos formas distintas de generación de modelos dependiendo del orden de los puntos del perímetro obtenidos del análisis de las estructuras blob que conforman la imagen de la firma. Un blob representa una estructura continua e identificable en una imagen, por ser de un color distinto del fondo. En

---

<sup>2</sup>Biometría es el estudio de métodos automáticos para el reconocimiento único de humanos basados en uno o más rasgos conductuales o rasgos físicos intrínsecos.

la primera de las formas los blobs son ordenados por tamaño y en la segunda por su orden natural de lectura, esto es de arriba abajo y de izquierda a derecha.

Las conclusiones de esta investigación demostraron que se obtiene mejores resultados con modelos de ordenación de los blobs de arriba-abajo y de izquierda a derecha.

### 1.3.3 Reconocimiento de caracteres

Al igual que en los casos anteriores existen numerosos ejemplos relacionados con el reconocimiento de caracteres [\[6\]](#) [\[7\]](#)

En [\[6\]](#) se presenta un algoritmo original y sencillo para el reconocimiento por regiones de caracteres manuscritos, en particular las cinco vocales. El algoritmo se basa en una estrategia por regiones en el dominio de la imagen. Los resultados de la evaluación preliminar del algoritmo propuesto, sobre una pequeña base de datos, mostraron una alta tasa de reconocimiento para las diferentes vocales. Con la finalidad de situar el algoritmo propuesto dentro del contexto de la arquitectura general de un sistema de reconocimiento de caracteres manuscritos, se hace una descripción de las funciones y de las tareas que realiza cada uno de los elementos que conforman dicha arquitectura, seguida de una breve explicación del diseño y su implementación. Sobre este trabajo se concluyó que el reconocimiento de la escritura manuscrita requiere del estudio de una gran cantidad de variables. Los resultados de aplicar este algoritmo arrojaron una tasa de reconocimiento de las vocales superior al 97% lo que hace al algoritmo propuesto un buen candidato para una aplicación real. Es importante mencionar que el algoritmo se probó con una pequeña base de datos y que restaría entonces validar los resultados con una base de datos más grande.

De los sistemas anteriormente citados, se puede observar que en todos ellos hay que extraer las características de las imágenes a estudiar y que esas características dependen del tipo de imagen que se esté tratando. Por tanto para el desarrollo de este proyecto se va a necesitar implementar algoritmos de manipulación de imágenes, por lo que es conveniente buscar

librerías adecuadas que contengan la mayor parte de las funciones que se van a utilizar, además por las características del proyecto en concreto, estas librerías deberían poder ser utilizadas desde diferentes entornos de programación y diferentes lenguajes. Por otro lado y puesto que es un proyecto de investigación es conveniente que sean de uso libre, altamente eficientes y de fácil utilización. Todas estas características las reúnen las librerías OpenCV<sup>3</sup> ya que son multiplataforma, existiendo versiones para Linux, Mac y Windows. Estas librerías incorporan más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión. Por otro lado están distribuidas bajo licencia BSD (*Berkeley Software Distribution*) que permite ser utilizadas libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

Se podría haber optado por programas de manipulación de imágenes como ImageJ<sup>4</sup>, se optó por OpenCV, por estar implementado en C++ lo cual permite una mayor rapidez de ejecución y mejor aprovechamiento de recursos ya que no utiliza la máquina virtual de Java.

Además de los algoritmos que nos permitan manipular las imágenes con comodidad y puesto que en el proyecto se intenta identificar qué características son las más apropiadas para la identificación de las imágenes se hace necesario añadir algún tipo de algoritmo de inteligencia artificial que nos ayude a identificar qué parámetros son los más adecuados para según qué imagen, es aquí donde entran las redes neuronales artificiales.

Para este proyecto se decidió, aconsejado por el tutor del proyecto, la utilización de redes neuronales backpropagation. Este tipo de redes transmiten los errores de la salida hacia atrás, hacia todas las neuronas, basándose en ese error recibido se reajustan los pesos de cada conexión de forma que, para ese mismo patrón, el error disminuya.

---

<sup>3</sup>OpenCV es una biblioteca de visión artificial desarrollada por Intel implementadas en lenguaje C y C++.

<sup>4</sup>ImageJ Programa de procesamiento de imágenes desarrollado en Java.



En nuestro caso, las redes neuronales nos ayudarán a, dados unos datos de entrada, que serán las características extraídas de las imágenes, indicaremos a qué imagen corresponden esos datos de entrada, de tal forma que al entrenar la red los pesos de las neuronas se vayan reajustando hasta obtener la salida deseada con el menor error posible.

Cuando las imágenes son complejas y no se sabe muy bien qué patrones o características son las que definen mejor dichas imágenes, la utilización de redes neuronales puede ayudar a averiguar estas características.

Lo ideal es tomar muchas imágenes de un solo tipo, en nuestro caso tomaremos imágenes de hojas de plantas, una vez calculadas las características se aplica un algoritmo de aprendizaje sobre las mismas. Uno de los métodos más sencillos consiste en tomar la distancia euclídea normalizada entre los valores de la misma clase. Con esto se pueden crear regiones en el hiper-espacio<sup>5</sup> para las clases diferentes. Debido a que la muestra de base de datos escogida cuenta con muy pocas imágenes de cada tipo, no se implementó ningún algoritmo de aprendizaje previo.

Es crucial la extracción de información de las imágenes, pero no cualquier información sino la información que se piense que más se adecúe a las imágenes tratadas. En nuestro caso se decidió que los mejores parámetros que a priori podrían ayudar más para la identificación de las imágenes propuestas, fueron los momentos invariantes de Hu, los cuales se calculan realizando una integral. En una imagen digital, por el hecho de ser discreta, la integral se convierte en un sumatorio en la que se le asignan diferentes pesos a cada píxel, dependiendo de la posición. Posteriormente se realizan operaciones sobre estos píxeles que permitan obtener un conjunto limitado de características. El objetivo es que estas características sean independientes a la rotación, traslación y escala. Al ser características que no varían significativamente respecto a estas variables se les ha dado el nombre de “momentos invariantes”.

---

<sup>5</sup>Un espacio de varias dimensiones

Además es necesario desarrollar algún mecanismo de comunicación entre el servidor y el teléfono móvil. Hay varias alternativas para llevar a cabo esto, una posibilidad es a través de un servicio web, esto es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Estas aplicaciones pueden estar hechas en distintos lenguajes y funcionar en distintas plataformas y sin embargo pueden comunicarse entre sí e intercambiar información mediante la utilización de dichos servicios. De entre las distintas alternativas que existen para comunicarse con un servicio web, cabe destacar dos:

**REST** es un tipo de servicio web que se asienta sobre el protocolo HTTP como mecanismo de transporte entre cliente y servidor, no impone ningún formato en concreto de datos, aunque lo más habitual es intercambiar información en formato XML<sup>6</sup> o JSON<sup>7</sup>. Actualmente lo implementan numerosas empresas como Ebay<sup>8</sup>, Amazon<sup>9</sup> o Yahoo<sup>10</sup>.

**SOAP** es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. También se asienta sobre el protocolo HTTP.

A pesar de que REST es uno de los más extendidos y utilizados actualmente, además de que es más ligero, con poca configuración, se lee fácilmente y no hace falta nada especial para implementarlo, en nuestro caso hemos decidido utilizar SOAP, puesto que es más aconsejable utilizar este protocolo en entornos donde se necesite un contrato formal y donde se describan todas las funciones de la interfaz.

---

<sup>6</sup> Lenguaje de marcas utilizado para guardar información en formato legible.

<sup>7</sup> Es un formato ligero para el intercambio de datos, formado por un subconjunto de la notación literal de objetos Java Script

<sup>8</sup> Sitio destinado a la subasta de productos.

<sup>9</sup> Compañía estadounidense de comercio electrónico.

<sup>10</sup> Compañía global de medios con sede en Estados Unidos.

Por otro lado toda la información que viaje a través del servicio web tendrá que estar almacenada en algún sitio. Es por esto que se utilizará una base de datos Microsoft SQL Server<sup>11</sup>, donde se guardará la información de las imágenes. Se podría haber optado por cualquier otro tipo de sistema gestor de base de datos como MySQL<sup>12</sup>, pero puesto que la mayor parte de los programas de reconocimiento de imágenes en el servidor se han desarrollado con el IDE Visual Studio<sup>13</sup> de Microsoft y dado que el servicio web está instalado y corriendo sobre Internet Information Service (IIS)<sup>14</sup> da menos problemas al permitir una mejor depuración de errores, y ser más amigable (mayor aporte de IntelliSense) en su programación.

### 1.4 Estructura del trabajo

Este trabajo está estructurado en 5 capítulos: Introducción, formado por cuatro partes, en donde se explica la motivación que ha dado lugar al desarrollo de esta investigación, los objetivos que se buscan en este trabajo, junto con el estado del arte y los elementos básicos para la realización del proyecto.

El siguiente capítulo explica toda la infraestructura del sistema implementado y junto con las herramientas y tecnología utilizadas.

El Capítulo 3 habla sobre el tratamiento realizado sobre las imágenes utilizadas como ejemplo y de la red neuronal diseñada para su reconocimiento.

---

<sup>11</sup>Sistema Gestor de Base de Datos creado y comercializado por Microsoft.

<sup>12</sup>Sistema Gestor de Base de Datos propiedad de Oracle Corporation.

<sup>13</sup>Entorno de desarrollo integrado desarrollado por Microsoft que permite programar en diferentes lenguajes.

<sup>14</sup>Es un servidor web y un conjunto de servicios desarrollado para el sistema operativo Microsoft Windows.

Para concluir se incluyen dos capítulos, más uno en el que se muestran los resultados obtenidos y otro en el que se exponen las conclusiones y el trabajo futuro.

## 2 Diseño del Proyecto

De la necesidad que surge de la identificación y clasificación de imágenes en teléfonos móviles basadas en el contenido de las mismas, y teniendo en cuenta lo comentado anteriormente, sería deseable crear una infraestructura genérica que sea capaz de reconocer imágenes de cualquier tipo con un coste mínimo de utilización de recursos. Por tanto, la solución planteada debe ser independiente del tipo de imagen a estudiar. Es por esto que se plantea la siguiente infraestructura para intentar resolver el problema.

### 2.1 Infraestructura del proyecto

La Figura 2.1.1, muestra el esquema utilizado para abordar el problema desde un prototipo real que permita el desarrollo de la investigación

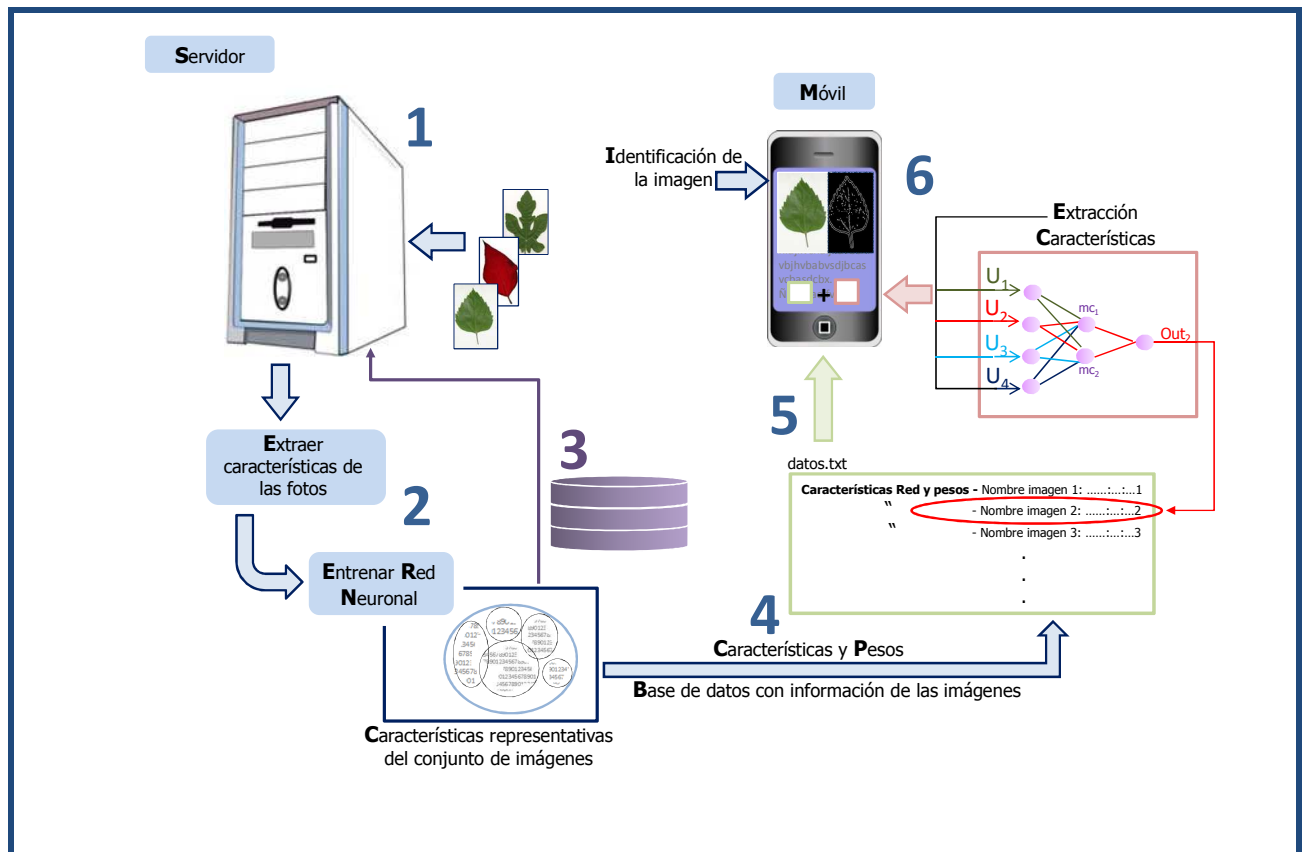


Figura 2.1.1

En esta figura 2.1.1 se representa el funcionamiento de la infraestructura creada. En primer lugar (1) se parte de unas imágenes capturadas con un teléfono móvil, junto con los datos descriptivos en un fichero de texto de las imágenes. Estas imágenes pueden ser de cualquier tipo que se pretenda estudiar, en nuestro caso utilizaremos las imágenes de unas hojas de plantas. Dichas imágenes serán almacenadas por el administrador del sistema en una carpeta del servidor. Tras un análisis del tipo de imagen a clasificar, se extraen las características más adecuadas de las mismas (2) y se procede a su clasificación. Se utilizarán Redes Neuronales para clasificar las imágenes ya que nos ayudarán tanto a clasificar las imágenes como a averiguar qué características son las más adecuadas para cada tipo de imagen. Esta Red Neuronal debe ser entrenada con las salidas de las imágenes clasificadas de forma que se obtenga el mínimo error posible.

Una vez entrenada, los datos ofrecidos por la red (pesos de sus neuronas, bias, número de capas ocultas, neuronas de entrada y de salida) junto con información adicional de cada imagen, (nombre, descripción, etc.), son almacenadas en una base de datos Microsoft SQL Server (3).

Por otro lado, un servicio web consumirá la información proveniente de la base de datos, y expondrá los datos en un servidor web (en nuestro caso se utilizó Internet Information Service IIS). Estos datos expuestos por el servicio web podrán ser consumidos vía SOAP por un teléfono móvil. (4).

Un programa desarrollado para Android<sup>15</sup> [\[ANEXO A.2\]](#), permitirá la captura de imágenes en un formato concreto. Hay que intentar que el front-end del usuario final ayude al reconocimiento de la imagen. En nuestro caso se desarrolló un aplicativo que buscaba los contornos mayores de la imagen. El programa de Android también permitirá acceder al servicio web para poder descargar la información de las imágenes a reconocer (5).

---

<sup>15</sup>Android, sistema operativo para dispositivos móviles desarrollado por Google.

Cuando un usuario quiera identificar qué tipo de objeto concreto es, realizará una foto de éste con el programa Android y se procederá a la extracción de sus características de forma similar a como se realizó en el servidor. Por último se utilizará un algoritmo que simulará el funcionamiento de la red neuronal y permitirá su clasificación definitiva (6).

### 2.2 Herramientas y tecnologías utilizadas

Para crear la infraestructura anterior se han tenido que utilizar diversas tecnologías y herramientas. Por un lado es necesario realizar un tratamiento de las imágenes tanto en el servidor como en el teléfono móvil, por las razones expuestas anteriormente se decidió utilizar las librerías OpenCV, por tanto, se tuvieron que instalar y configurar tanto en Visual Studio [\[ANEXO A.3\]](#) como en Android estas librerías.

El hecho de haber implementado el programa de Visual Studio utilizando el lenguaje C++ y compilado en modo “*Common Language Runtime Support (/clr)*”<sup>16</sup>, permitió utilizar las librerías nativas de OpenCV directamente, sin necesidad de utilizar un wrapper o envoltorio que aísle la complejidad de trabajar con C++, por lo que se gana en rapidez y eficiencia. Si hubiésemos utilizado otro lenguaje como C#, habría sido más sencilla la programación pero en contrapartida habría sido menos eficiente.

Por otro lado en Android se utilizó el wrapper de OpenCV para java. Se podrían haber utilizado directamente las librerías nativas de OpenCV, y a través del JNI<sup>17</sup> acceder a las librerías nativas, pero apenas se gana en eficiencia y la dificultad es mayor.

Por otro lado es necesario diseñar una infraestructura de comunicación entre el servidor y el dispositivo móvil. Para ello se decidió crear un servicio web que accediera a la información

---

<sup>16</sup>Modo de compilación mixto capaz de manejar tanto instrucciones máquinas como instrucciones administradas.

<sup>17</sup>Framework que permite que un programa escrito en Java y ejecutado en la máquina virtual de Java pueda interactuar con programas escritos en otros lenguajes como C++.

guardada en la base de datos y que permitiera la comunicación entre ambas partes. Este servicio fue desarrollado en lenguaje C# puesto que iba a alojarse sobre un servidor web Internet Information Service o IIS. Este servidor web viene por defecto en el sistema operativo Windows 7 profesional y su configuración es muy sencilla. Se podría haber utilizado otros servidores web como Apache Tomcat<sup>18</sup>, pero éste está diseñado para lenguaje Java y el programa se realizó en C#.

Como se explicó anteriormente la comunicación entre el servicio web y el teléfono móvil se realizará utilizando el protocolo de comunicación SOAP, puesto que este ofrece una filosofía de contrato entre ambas partes y esto se ajusta más que si hubiésemos implementado una comunicación de tipo REST.

La información transmitida entre ambas partes se guarda en una base de datos en el servidor.

---

<sup>18</sup>Servidor web de código abierto



A continuación se muestra el diseño de la base de datos elegido para esta aplicación.

### Diagrama de Int./Relación:

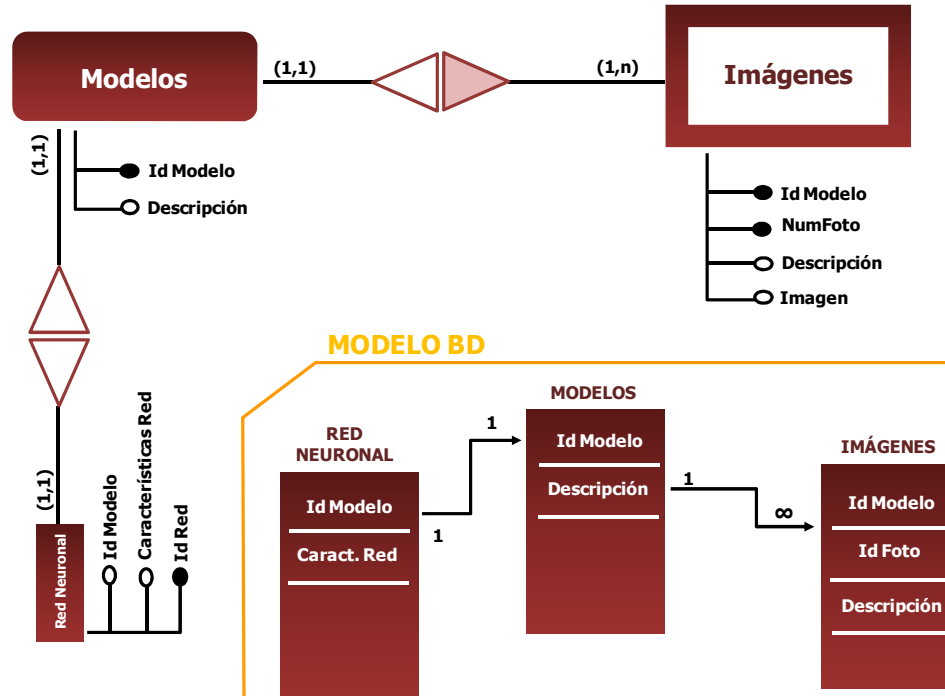


Figura 2.2.1 Modelo Base de Datos

El modelo elegido está formado por tres tablas:

- **Red\_Neuronal:** Contiene la información de las redes neuronales de cada clase de imagentratada.
- **Modelos:** tienela información de las clases de imágenes tratadas.
- **Imágenes:**almacenará los datos de las imágenes tratadas

La tabla **Red\_Neuronal** se relaciona en una cardinalidad de 1:1 con la tabla **Modelos** puesto que una red neuronal pertenecerá a un solo modelo o clase de imágenes. Y un modelo de imagen solo tendrá asociada una red neuronal.

La tabla **Modelos** se relaciona con la tabla **Imágenes** con una cardinalidad de 1: N puesto que un modelo de imagen podrá tener una o muchas imágenes pertenecientes a ese modelo.

Por ejemplo el modelo de imagen “zapatillas” [\[ANEXO A4\]](#) tendrá muchas Imágenes asociadas a ese modelo.

Para identificar las imágenes en la tabla imágenes, hacemos uso de un atributo discriminador “NumFoto” junto con “IdModelo”, se ha optado por este modelo puesto que parece más natural que una imagen no pueda existir en nuestra base de datos sino pertenece a un modelo concreto.

Además el campo Imagen contiene la imagen, pero esta no se envía al móvil. Se introdujo por si es necesario enviarla al móvil en algún momento.

Por último para poder realizar esta clasificación es necesario crear un programa que entrene la red neuronal para reconocer las imágenes, este programa estará implementado en Matlab<sup>19</sup>, el cual leerá de un fichero las características numéricas de las imágenes, junto con su tipo y los introducirá en una red neuronal, la entrenará de forma supervisada hasta la conseguir la convergencia de la red y en ese momento será capaz de, para un vector de características de entrada averiguar el tipo correspondiente al que pertenece.

---

<sup>19</sup>Es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE), con un lenguaje de programación propio (lenguaje M)

### 3 Tratamiento de imagen

El primer paso para conseguir nuestros objetivos es obtener las características que nos permitan diferenciar entre las distintas clases de imágenes almacenadas, en nuestro caso utilizaremos una base de datos de 25 imágenes de hojas de diversas plantas. Las imágenes tratadas en el servidor serán obtenidas con la aplicación que se creará en Android, para que sean lo más parecidas posibles a las imágenes que luego se intentarán identificar. Además la aplicación del dispositivo ayudará al usuario, a la correcta obtención de las imágenes y de esta forma se puedan preparar para una correcta extracción de la información de las mismas. Sin embargo en el teléfono móvil las fotos pueden venir con diferente calidad, iluminación, ángulo, escala, etc. Por lo que los algoritmos de extracción deberán ser lo más parecidos posibles sino iguales en ambas plataformas (servidor y móvil). Para la correcta obtención de las características de las imágenes se realizarán diferentes operaciones sobre cada tipo de imagen. Estas operaciones podrán ser distintas dependiendo del tipo de imagen estudiado. En las siguientes secciones se describirán las etapas de procesamiento que se realizaron en este proyecto.

#### 3.1 Escala de Grises

Primero la imagen debe convertirse en un formato más apropiado para extraer las características. Se consideró que en esta primera aproximación, que lo más relevante de las imágenes a tratar era su contorno, y para hallar su contorno no es necesario tener en cuenta el color de las imágenes. Por lo que es más conveniente convertir la imagen a una escala de grises. En este proyecto se trabajó con el modelo RGB. En este modelo se asigna una intensidad a cada uno de los tres colores primarios de luz: rojo, verde y azul, de esta manera cada píxel en una fotografía se representa mediante un valor que identifique la intensidad de cada uno de estos tres colores que, mezclados por adición, se acerque más al color verdadero del píxel, las cámaras digitales modernas utilizan valores entre 0 y 255 ( $2^8$ ) permitiendo de esta manera más de 16 millones de colores distintos. En una escala de grises en cambio, cada píxel es representado con un único valor entre 0 y 255, el cual representa que tan oscuro es el mismo. Para convertir un píxel de una imagen en color a uno en escala de grises se suele

hacer un promedio ponderado de la intensidad de cada uno de los tres colores en donde a cada color se le asigna un peso. Al ser un promedio ponderado la suma de los tres pesos debe ser igual a 1. Los pesos utilizados por OpenCV son:

$$gris = 0,2989 * rojo + 0,5870 * verde + 0,1140 * azul$$

Estos pesos fueron calculados a partir de observaciones de la sensibilidad del ojo humano a cada uno de los tres colores.

El resultado de aplicar la anterior fórmula es el siguiente:



Figura 3.1.1 Imagen original



Figura 3.1.2 Imagen escalada a grises

### 3.2 Filtrado de imagen

Es necesario filtrar la imagen para quitar ruido, para realizar esto es necesario reducir la amplitud de las variaciones de la imagen, una forma simple de hacer esto, es reemplazar cada píxel por la media del valor de los píxeles de alrededor, de esta forma las variaciones rápidas de intensidad pueden ser suavizadas y reemplazadas por una transición más gradual. La función de OpenCV elegida para realizar este suavizado ha sido GaussianBlur, esta función tiene en cuenta el peso de los píxeles más cercanos que los alejados. La función realizada por este algoritmo de suavizado es la siguiente<sup>20</sup>:

$$G(x, y) = \frac{1}{2 * \pi * \sigma^2} * e^{\frac{-x^2 + y^2}{2 * \sigma^2}}$$

Donde la “x” es la distancia desde el origen (punto del píxel a tratar) en el eje horizontal, e “y” es la distancia desde el origen en el eje vertical y “σ” es la desviación estándar de la

---

<sup>20</sup>Fuente [http://en.wikipedia.org/wiki/Gaussian\\_blur](http://en.wikipedia.org/wiki/Gaussian_blur)

distribución Gaussiana. Cuando se aplica en dos dimensiones, esta fórmula produce una superficie cuyos contornos son círculos concéntricos con la distribución Gaussiana desde el punto central.

Si aplicamos la función `GaussianBlur` de OpenCV sobre la imagen escalada a grises, obtenemos el siguiente resultado:



Figura 3.2.1 Imagen escalada a grises



Figura 3.2.2 Imagen suavizada

### 3.3 Binarización

Una vez obtenida la imagen en escala de grises y después de filtrarla se procedió a hacer una binarización. La binarización es un proceso que transforma una imagen que está en escala de grises a una imagen en dos colores: blanco y negro. Esto es útil para el reconocimiento de las hojas porque éstas se distinguen principalmente por la forma de la hoja mucho más que por la intensidad de los píxeles que los componen. De esta forma, al convertir las imágenes en blanco y negro se pierde muy poca información relevante y se simplifica la representación de cada píxel, pasando de 256 posibles valores a tan solo 2. Un método muy utilizado para binarizar una imagen es escoger un valor umbral  $T(x, y)$  para cada píxel  $(x, y)$  y establecer el valor de cada píxel como **THRES\_BINARY**:

$$dst(x, y) = \begin{cases} 0 & \text{si } I(x, y) < T(x, y) \\ 255 & \text{en cualquier otro caso} \end{cases}$$

La ecuación anterior se implementó en el servidor y Android, sin embargo cuando las imágenes tienen poca luz, se comprobó que se obtenía mejor el contorno de la misma con la siguiente variación **THRES\_BINARY\_INV**:

$$dst(x, y) = \begin{cases} 255 & \text{si } I(x, y) < T(x, y) \\ 0 & \text{en cualquier otro caso} \end{cases}$$

Donde  $I(x, y)$  es la imagen original y  $dst(x, y)$  es la imagen binarizada. Normalmente, la diferencia entre los distintos métodos de binarización se encuentra en cómo elige cada uno el valor umbral  $T(x, y)$ .

El resultado obtenido sobre la imagen suavizada es el siguiente:



Figura 3.3.1 Imagen suavizada

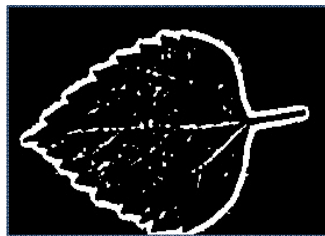


Figura 3.3.2 Imagen Binarizada

Dependiendo de la imagen el valor de umbralización puede variar para conseguir el efecto deseado, si la imagen proviene de un escáner, la iluminación es constante, pero en este caso tanto las imágenes tratadas en el servidor como las imágenes provenientes de la cámara del móvil tienen iluminaciones no tan constantes, por lo que se decidió elegir un método de binarización local adaptativo. En un método de este tipo  $T(x, y)$  puede ser distinto para cada píxel, y se dice que es local porque  $T(x, y)$  se calcula utilizando información sobre el vecindario de píxeles del píxel  $(x, y)$ .

### 3.4 Detección de bordes (Canny)

Antes de buscar el contorno se decidió aplicar otro filtro más, esta vez con la intención de detectar bordes, este filtro es opuesto al filtro GaussianBlur, esta vez buscamos, no suavizar la imagen para quitar ruido, sino todo lo contrario buscar los bordes y formas de la imagen.

El algoritmo Canny fue desarrollado por John F. Canny en 1986, es un algoritmo de múltiples etapas para detectar una amplia gama de bordes en imágenes. El propósito de Canny era descubrir un algoritmo óptimo de detección de bordes. Para que un detector de bordes pueda ser considerado óptimo debe cumplir los siguientes puntos:

- Buena detección, el algoritmo debe marcar el mayor número real en los bordes de la imagen como sea posible.
- Buena localización, los bordes de marca deben estar lo más cerca posible del borde de la imagen real.
- Respuesta mínima, El borde de una imagen sólo debe ser marcado una vez, y siempre que sea posible, el ruido de la imagen no debe crear falsos bordes.

El algoritmo de Canny pasa por dos etapas: reducción de ruido y encontrar la intensidad del gradiente de la imagen.

A continuación se muestra como se quedan las imágenes una vez aplicado el algoritmo de Canny.

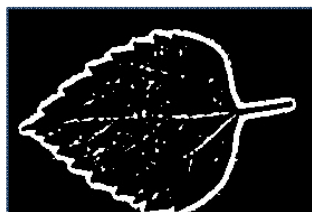


Figura 3.4.1 Imagen Binarizada



Figura 3.4.2 Imagen Canny

### 3.5 Detección de contorno

El principal objetivo que persiguen los pasos anteriores es la detección de objetos dentro de la imagen. En OpenCV existe un método que detecta contornos, en los que dada una imagen de entrada binarizada ofrece de salida un vector de contornos, y cada contorno es un vector de puntos. Puesto que una imagen puede contener varios contornos, en la imagen tomada con

la cámara del móvil, se decidió tratar solamente el contorno mayor. Uno de los parámetros del método findContours es **CV\_CHAIN\_APPROX\_NONE**, con esta opción obtenemos todos los puntos del contorno.

Añadir que si se quiere aislar y dibujar la figura con contorno mayor del resto. Existe un método llamado drawContour que dibuja el contorno, pero además la opción thickness si se indica un -1, dibuja todo el interior del contorno del color elegido.



Figura 3.5.1 Imagen Canny

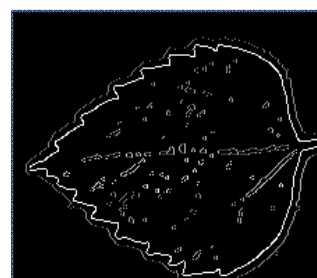


Figura 3.5.2. Imagen contorno

### 3.6 Extracción de características

Finalmente, después de procesar adecuadamente la imagen se obtienen las características más significativas que permita diferenciar los objetos a clasificar, independientemente del tamaño o inclinación que tenga la hoja en cuestión. Si las imágenes fueran adquiridas mediante un escáner, la inclinación podría ser muy pequeña, incluso tanto que podría despreciarse. Sin embargo, cuando las imágenes son adquiridas con un móvil, la inclinación entra a ser un factor importante. Por esta razón es necesario encontrar características que sean invariantes a la rotación y a la escala, es decir, que al rotar o escalar una imagen, el valor numérico de las características sea similar.

Para lograr un cierto grado de independencia de las características a la rotación y a la escala se pueden tomar dos enfoques: un primer enfoque es transformar la imagen, esto significa rotarla y escalarla para hacerla lo más similar posible a un patrón estándar, después de esto se pueden extraer características numéricas de la imagen. Evidentemente en cualquiera de los



dos casos, las características buscadas deben ser lo más similares posibles para objetos de la misma clase y lo más diferentes posibles para objetos de distintas clases. Un segundo enfoque consiste en trabajar con la imagen tal cual es tomada por la cámara, y utilizar características en las que se obtengan valores numéricos similares para una misma imagen rotada y escalada, es decir que sean invariantes a la rotación y a la escala.

### 3.6.1 Rotación y escala mediante transformación geométrica

Si se utilizan características dependientes a la rotación, se debe entrenar el clasificador (red neuronal Backpropagation) con distintas instancias que tengan distintos ángulos de rotación. Si se hace difícil conseguir ejemplos reales, una opción puede ser construir ejemplos virtuales. El problema con este enfoque es que tiene una alta complejidad computacional y que instancias de una misma clase pueden quedar ampliamente repartidas por el hiperespacio de clasificación. Como resultado sería más difícil para un algoritmo de clasificación separar las distintas clases.

Un enfoque más práctico es detectar la rotación y la escala de la imagen, y posteriormente aplicar transformaciones geométricas para ajustar la rotación y la escala a valores preestablecidos. Después de esto se podrían obtener características dependientes a la rotación y a la escala para detectar la hoja en imagen.

La escala se podría ajustar haciendo una interpolación de los píxeles al tamaño deseado. Existen, sin embargo, dos problemas con esta solución. En cuanto a la rotación, si el ángulo  $\theta$  no es un múltiplo de  $90^\circ$ , las coordenadas de los nuevos píxeles  $(x', y')$  no serían enteras. Esto se puede solucionar interpolando  $(x', y')$  a sus vecinos enteros más cercanos, pero este proceso conlleva una pérdida de información importante cuando la calidad de las imágenes no es muy buena. En cuanto a la escala, la interpolación también produce una pérdida de información.

Para el reconocimiento de hojas de plantas es conveniente encontrar métodos de extracción de características que minimicen la pérdida de información, pero que al mismo tiempo sean invariantes a la rotación y a la escala. Los momentos invariantes, como los momentos de Hu [8] o de Flusser [9], son características que cumplen dicha propiedad.

### 3.6.2 Momentos invariantes de Hu

Los momentos invariantes fueron propuestos por primera vez por Hu. Estos momentos pueden ser considerados como un promedio ponderado de los píxeles de una imagen. Hu computo sus invariantes utilizando los momentos geométricos, los cuales son variantes a la rotación y a la escala. Los momentos geométricos se definen como:

$$\mu_{pq} = \iint (x - \bar{x})^p (y - \bar{y})^q f(x, y) dy dx$$

Donde  $\mu_{pq}$  es el momento geométrico de orden  $(p + q)$ ,  $f(x, y)$  es el valor del píxel en la posición  $(x, y)$  de la imagen y  $(\bar{x}, \bar{y})$  es el centroide de la misma. Partiendo de estos momentos podemos obtener  $n_{pq}$ , un momento de orden  $(p + q)$  que es invariante a la escala:

$$\mu_{pq} = \frac{\mu_{pq}}{\mu_{00}^{1+\frac{p+q}{2}}}$$

Utilizando  $\mu_{pq}$ , Hu logro un conjunto de momentos invariantes, todos con un grado menor o igual a 3. Sus siete momentos invariantes son:

$$\begin{aligned}\phi_1 &= n_{20} + n_{02}, \\ \phi_2 &= (n_{20} + n_{02})^2 + 4n_{11}^2, \\ \phi_3 &= (n_{30} + 3n_{12})^2 + (3n_{21} - n_{03})^2, \\ \phi_4 &= (n_{30} + n_{12})^2 + (n_{21} + n_{03})^2,\end{aligned}$$

$$\begin{aligned}\emptyset_5 &= (n_{30} - 3n_{12})(n_{30} + n_{12})((n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2) + (3n_{21} - n_{03})(n_{21} \\ &\quad + n_{03}) \times (3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2), \\ \emptyset_6 &= (n_{20} - n_{02})((n_{30} + n_{12})^2 - (n_{21} + n_{03})^2) + 4n_{11} (n_{30} + n_{12})(n_{21} + n_{03} \\ \emptyset_7 &= (3n_{21} - n_{03})(n_{30} + n_{12})((n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2) - (n_{30} - 3n_{12})(n_{21} \\ &\quad + n_{03}) \times (3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2)\end{aligned}$$

### 3.7 Obtención red neuronal

Una vez extraídas las características de las imágenes conocidas que tenemos almacenadas en la Base de Datos, debemos encontrar un sistema que dado un conjunto de características de una imagen nos diga a qué imagen de las almacenadas más se parece o corresponde. Esto es lo que se conoce como clasificador o reconocedor a los datos. La utilización de redes neuronales para clasificar datos es una buena opción como demuestran multitud de ejemplos existentes en la literatura [\[10\]](#) [\[11\]](#).

Por las características del proyecto se decidió utilizar un tipo de red neuronal Backpropagation.

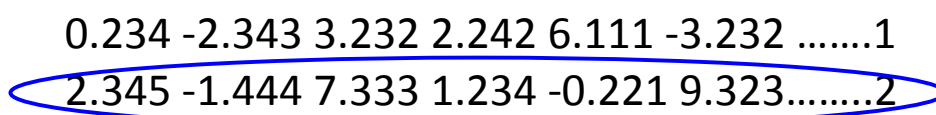
Entre las ventajas que tienen este tipo de redes, es que aprovecha la naturaleza paralela de las redes neuronales para reducir el tiempo requerido por un procesador secuencial para determinar la correspondencia entre unos patrones dados. Por otro lado hay que reconocer qué patrones son los más adecuados, es decir, de todos los parámetros enviados a la red unos serán más significativos que otros. Las redes neuronales ayudarán a encontrar los parámetros más significativos.

Una vez que se ha aplicado un patrón de entrada a la red como estímulo, este se propaga desde la primera capa al resto de capas de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

En nuestro caso se abordó la clasificación por parte de la red neuronal desde dos perspectivas. Por un lado en el servidor, a través de un programa creado en Matlab, este programa recibirá como entrada los parámetros o características de las imágenes, en nuestro caso los momentos Hu.

Por otro lado se estudiará la clasificación que realiza el móvil, implementando un algoritmo de simulación de la red neuronal (figura 4.1.10) con la ayuda de la información recibida desde el servicio web.

Por el lado del servidor y una vez entrenada la red se obtendrán las características de la misma para ser almacenadas en la base de datos. En la figura 3.7.1 se puede observar un ejemplo del fichero que toma el programa Matlab.



```
0.234 -2.343 3.232 2.242 6.111 -3.232 .....1
2.345 -1.444 7.333 1.234 -0.221 9.323.....2
```

Figura 3.7.1

Los primeros datos son las entradas a la red neuronal, en nuestro caso momentos Hu del contorno de las imágenes. Los valores 1, 2, etc. son los targets u objetivos también introducidos en la red. Cuando se entrena la red, los pesos de la red neuronal se van ajustando para poder producir esas salidas. Puesto que es una red supervisada, cuando no se produce la salida deseada, se propaga un error para ir modificando los pesos y así conseguir la salida deseada para esas entradas.

Es necesario estudiar el error producido, esto significa que para unos datos de entrada, cuanto se parece lo obtenido de lo que debería haber clasificado. Se hicieron pruebas intentando minimizar al máximo el error. La figura 3.7.2 muestra la evolución del error.

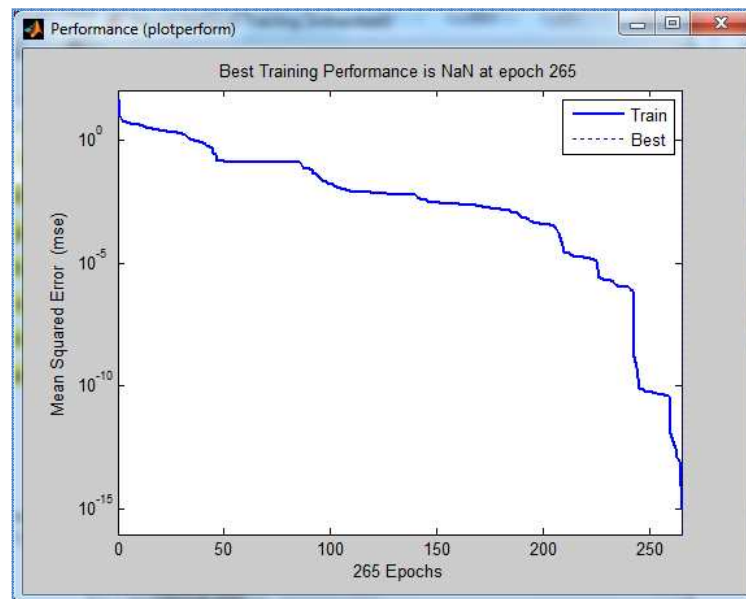


Figura 3.7.2

Como se puede observar, el error de la red disminuye a valores muy bajos en muy pocas iteraciones.

Se decidió utilizar la función “purelin<sup>21</sup>” para la activación de las neuronas de la capa de salida, puesto que se pretende que las salidas no estén acotadas entre ningún valor. No obstante en la capa oculta se pueden utilizar otros valores.

Como función de entrenamiento de la red se decidió utilizar “trainscg”, esta función entrena redes multicapa con retropropagación actualizando los pesos y bias, de acuerdo al método del gradiente conjugado, que permite una mayor rapidez de convergencia.

A continuación se muestra parte esencial del código utilizado (Figura 3.7.3)

---

<sup>21</sup>Función lineal pura, es decir la red es la salida.

```
datos = load ('fichMatlab.txt');%cargamos los datos de
entrada
p=datos(1:20,1:7);
t=datos(1:20,8);
net=newff(p',t',2,{ 'purelin', 'purelin'}, 'trainscg'); (1*)
net.divideFcn='';
net.inputs{1}.processFcns={}; (2*)
net.outputs{2}.processFcns={}; (3*)
net.trainParam.goal=0;
net.trainParam.epochs=5000;
net=train(net,p',t');
```

Figura 3.7.3

Se puede observar que en la capa de salida se utilizó la función lineal “purelin” (1\*), además es necesario que las entradas y las salidas no tengan ninguna función de procesamiento de datos (2\*) y (3\*). Esto se tuvo que hacer porque al aplicar en el teléfono móvil el algoritmo de simulación (Figura 4.1.10), no se obtenían los resultados esperados, debido a que Matlab por defecto pre procesa los datos, normalizándolos.

## 4 Evaluación de resultados

Una vez extraídas las características de las imágenes, vamos a comprobar si estas características son las adecuadas para conseguir nuestro objetivo. Primero realizaremos un estudio sobre los parámetros más adecuados que debe tener la red neuronal de forma que obtengamos un buen clasificador y por otro lado estudiaremos la viabilidad del sistema propuesto desde el punto de vista del usuario.

### 4.1 Estudio de la red neuronal

Para realizar el estudio se parte de una base de datos formada por 25 imágenes de hojas de 12 tipos de planta distintas. Todas las fotos fueron sacadas con la cámara de la aplicación desarrollada para el teléfono móvil, exceptuando las 4 primeras imágenes que se sacaron con otro teléfono para variar la procedencia de las imágenes. El teléfono móvil utilizado fue un LG-E400 con versión de Android 2.3.6.

El siguiente gráfico muestra la distribución de las imágenes en la base de datos.

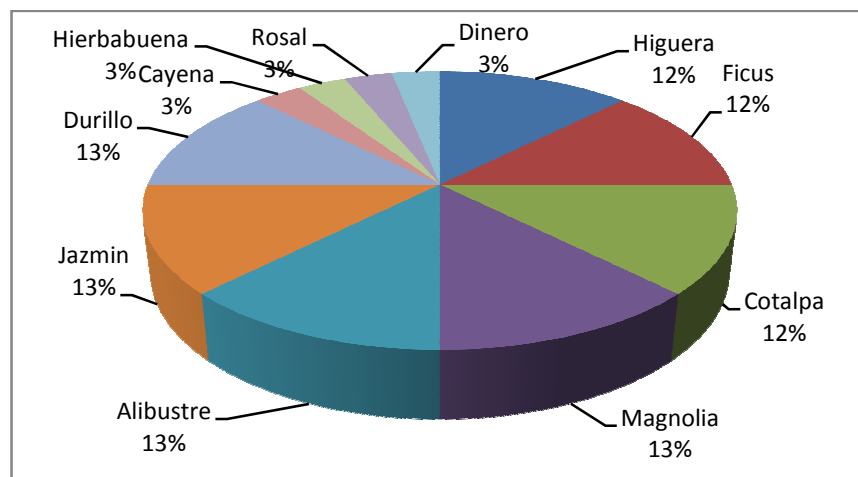
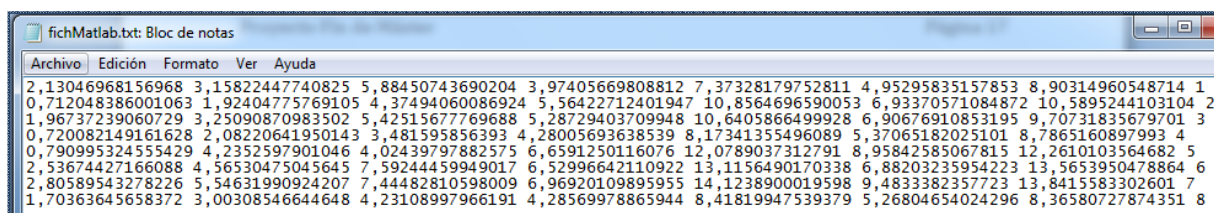


Figura 4.1.1

De ellas se escogieron 18 fotos, el 80% para crear el grupo de entrenamiento de la red neuronal. Las otras 7 (correspondientes a Higuera, Ficus, Catalpa, Magnolia, Aligustre, Jazmín, Durillo), el 20%, se utilizaron para el conjunto de test.

El proceso fue el siguiente:

1.- Se obtuvieron los momentos Hu de las imágenes de las plantas anteriores, en la figura 4.1.2 se muestra un ejemplo de los momentos Hu para las 8 primeras imágenes.



fichMatlab.txt: Bloc de notas																
Archivo Edición Formato Ver Ayuda																
2,13046968156968	3,15822447740825	5,88450743690204	3,97405669808812	7,37328179752811	4,95295835157853	8,90314960548714	1									
0,712048386001063	1,92404775769105	4,37494060086924	5,56422712401947	10,8564696590053	6,93370571084872	10,5895244103104	2									
1,96737239060729	3,25090870983502	5,42515677769688	5,28729403709948	10,6405866499928	6,90676910853195	9,70731835679701	3									
0,720082149161628	2,08220641950143	3,481595856393	4,28005693638539	8,17341355496089	5,37065182025101	8,7865160897993	4									
0,790995324555429	4,2352597901046	4,02439797882575	6,6591250116076	12,0789037312791	8,95842585067815	12,2610103564682	5									
2,53674427166088	4,56530475045645	7,59244459949017	6,52996642110922	13,1156490170338	6,88203235954223	13,5653950478864	6									
2,80589543278226	5,54631990924207	7,44482810598009	6,96920109895955	14,1238900019598	9,4833382357723	13,8415583302601	7									
1,70363645658372	3,00308546644648	4,23108997966191	4,28569978865944	8,41819947539379	5,26804654024296	8,36580727874351	8									

Figura 4.1.2 Base de datos

2.- Después la aplicación de Matlab lee estos datos, los inyecta a la red neuronal, entrena la red y escribe los pesos y bias (características de la red una vez entrenada) en un fichero llamado “salida.txt”.

3.- Los resultados, junto con la información relacionada para cada planta son almacenados en la base de datos. La figura 4.1.3 presenta una captura de la base de datos.



DatosImagen
-0.165853 1.550495 0.129380 -0.018481 1.15410...

Figura 4.1.3 Base de datos

Los datos que aparecen en el campo “DatosImagen” corresponden a los pesos y bias de la red neuronal una vez entrenada, es decir, es la salida del programa Matlab. Nótese que estos



pesos y bias siempre son los mismos para todas las imágenes, por lo que se guardan en la tabla Red\_Neuronal una sola vez por cada clase de imagen estudiada.

Se realizaron varias pruebas para estudiar la convergencia de la red, es decir, cuando los pesos de las neuronas minimizaban los errores de salida. Como los datos de entrada eran los 7 momentos de Hu, la arquitectura de la red debía tener 7 neuronas de entrada. Para averiguar el número de neuronas de la capa oculta se realizaron diversas pruebas, el número que mejores datos ofreció fue de 2 neuronas en la capa oculta. Como puede observarse en la figura 4.1.4, el error disminuía considerablemente en no demasiadas iteraciones con 2 neuronas en la capa oculta.

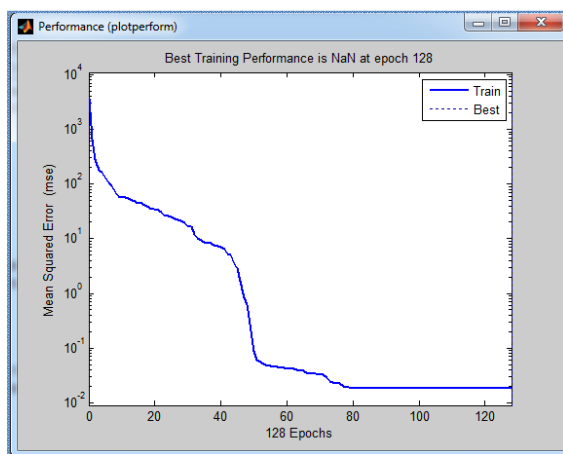


Figura 4.1.4

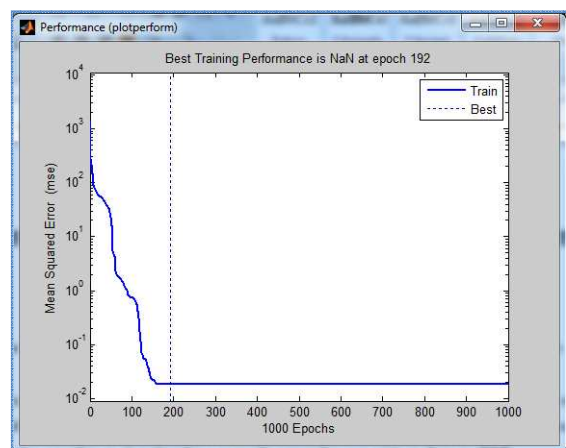
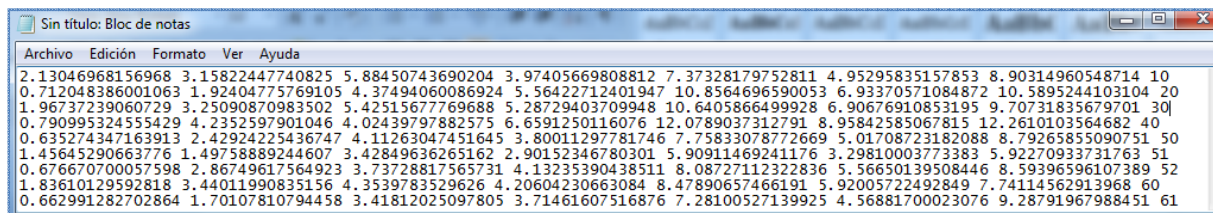


Figura 4.1.5

Con tres neuronas en la capa oculta los resultados no variaban demasiado pero se tardaba un poco más en conseguir la convergencia (192 epoch) (figura 4.1.5).

Se observó que al agrupar valores de salidas separados de 10 en 10 (cada grupo de 10 se refiere a los valores de salida referidos a las imágenes tomadas de una misma planta) se obtenían mejores resultados. La figura 4.1.6 muestra la forma en la que se distribuyeron los valores de salida.



Sin título: Bloc de notas

Archivo Edición Formato Ver Ayuda

2.13046968156968	3.15822447740825	5.88450743690204	3.97405669808812	7.37328179752811	4.95295835157853	8.90314960548714	10
0.712048386001063	1.92404775769105	4.37494060086924	5.56422712401947	10.8564696590053	6.93370571084872	10.5895244103104	20
1.96737239060729	3.25090870983502	5.42515677769688	5.28729403709948	10.6405866499928	6.90676910853195	9.70731835679701	30
0.790995324555429	4.2352597901046	4.02439797882575	6.6591250116076	12.0789037312791	8.95842585067815	12.2610103564682	40
0.635274347163913	2.42924225436747	4.11263047451645	3.80011297781746	7.75833078772669	5.01708723182088	8.79265855090751	50
1.45645290663776	1.49758889244607	3.42849636265162	2.90152346780301	5.90911469241176	3.29810003773383	5.92270933731763	51
0.676670700057598	2.86749617564923	3.73728817565731	4.13235390438511	8.08727112322836	5.56650139508446	8.59396596107389	52
1.83610129592818	3.44011990835156	4.3539783529626	4.20604230663084	8.47890657466191	5.92005722492849	7.74114562913968	60
0.662991282702864	1.70107810794458	3.41812025097805	3.71461607516876	7.28100527139925	4.56881700023076	9.28791967988451	61

Figura 4.1.6

Por otro lado, se probó en cambiar la función de transferencia de la capa oculta por una “tansig<sup>22</sup>”, el resultado fue que mejoró la convergencia y se utilizaron menos epoch (figura 4.1.7)

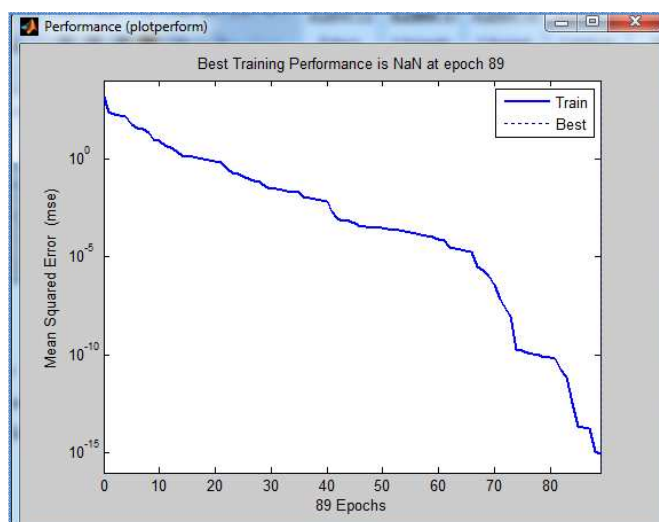


Figura 4.1.7

Tras este estudio se puede concluir que la red más adecuada de nuestro sistema debe estar formada por 7 neuronas de entrada, 2 neuronas en la capa oculta con función de transferencia “tansig” y una neurona de salida con función de transferencia “purelin”.

---

<sup>22</sup>Función tangente sigmoide.

Además hay que comprobar el porcentaje de imágenes que es capaz de clasificar correctamente nuestra red.

Para realizar esto, utilizaremos las imágenes del grupo de test y comprobaremos como las clasifica nuestra red.

Obviamente si utilizamos las imágenes del servidor de la batería de imágenes de entrenamiento los valores obtenidos son los esperados. Se realizó una prueba para confirmarlo.

La figura 4.1.8 muestra la imagen de la hoja de hierbabuena del conjunto de imágenes que se utilizó para entrenar la red.



Figura 4.1.8 Hoja Hierbabuena

Pero si utilizamos imágenes del conjunto de test, puede ser que los resultados no sean los esperados.

Por ejemplo, para la siguiente imagen (figura 4.1.9)



Figura 4.1.9 Hoja Higuera

Los Momentos Hu ofrecidos para esta hoja fueron:

0.635274347163913 2.42924225436747 4.11263047451645 3.80011297781746 7.75833078772669 5.01708723182088 8.79265855090751

Para comprobar si se ha identificado la imagen. Es necesario comprobar que estos datos (momentos Hu), indican que se refieren a la planta 5 (hierbabuena), para hacer esto hay dos opciones:

- Introducirlos en la función “Sim” de Matlab, que permite simular que dados unos datos de entrada se ofrece un valor de salida, que en este caso debería ser 5 o estar cercano al 5.
- En Android no disponemos de Matlab, por lo que es necesario implementar la función “Sim” de Matlab, ayudándonos de los pesos y bias recibidos del servidor, podemos implementar la función como se describe en la figura 4.1.10

**REALIZAMOS EL SIM DE LA RED:**

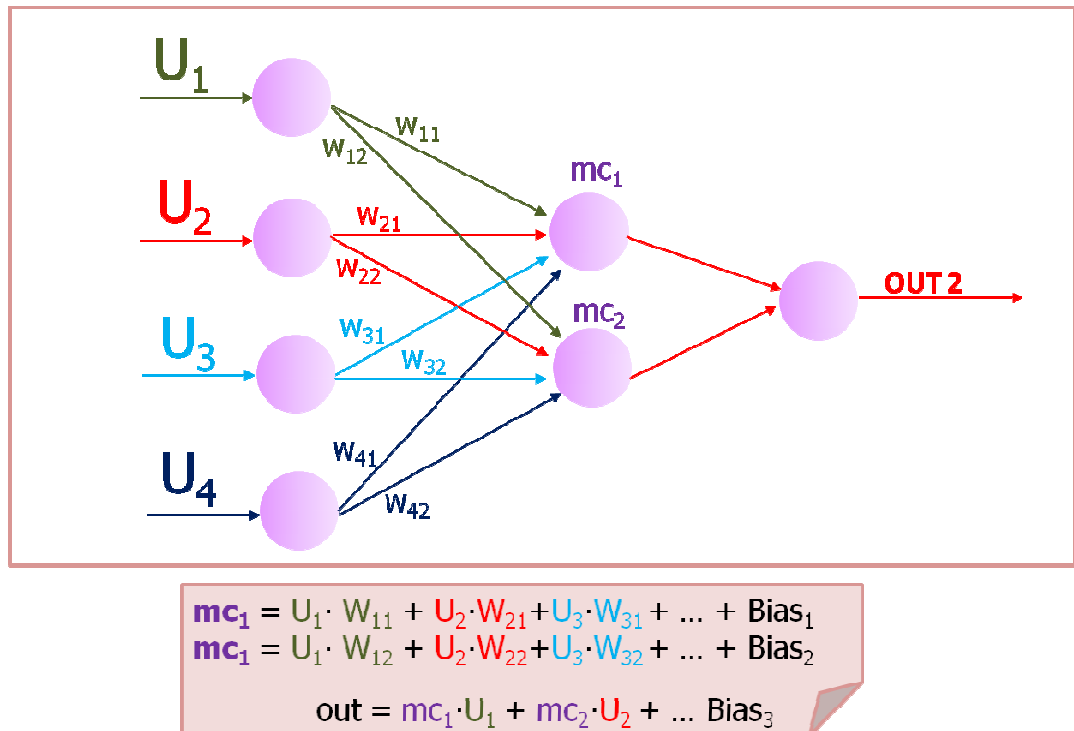


Figura 4.1.10 función Sim

Si queremos comprobar que los datos se refieren a la planta 5 por la primera opción (utilizando la función “Sim” de Matlab, se realiza lo siguiente: si “p” son los momentos Hu de todas las imágenes tratadas, net es la red neuronal y el 5 hace referencia a la quinta fila de “p”, y los introducimos en la función “Sim” de Matlab de la siguiente forma:

`Sim(net,p(5,:))`

Obtenemos como resultado el **4,879** valor cercano al **5** que buscábamos.

Para la siguiente prueba se utilizaron dos hojas de “Cotalpa”, una de ellas (figura 4.1.11) pertenece al conjunto de imágenes que se entrenaron, la otra (figura 4.1.12) pertenece al conjunto de imágenes de test.



Figura 4.1.11 Hoja Cotalapa



Figura 4.1.12 Hoja Cotalpa test

En este caso las dos imágenes son distintas, además una está girada 90° respecto de la otra. Obtenemos los momentos Hu de la primera imagen.

### Primera imagen

1.94800848938158 3.55570781907673 5.01286680220139 4.06673877090947 7.84735333548514 5.21419269315903 8.59985742796143

### Segunda imagen

1.9363631626116 3.54296871709557 4.68330097936299 4.19511307266228 8.38142975693312 5.7855620720928 8.55309964075296

Podemos observar que son valores parecidos. Si aplicamos la función “Sim” a los momentos Hu de la segunda imagen, obtenemos: **65.9797** que es un valor cercano al valor esperado **67.4927**.

Consideremos esta vez las dos siguientes imágenes:



Figura 4.1.13 Hoja Yedra

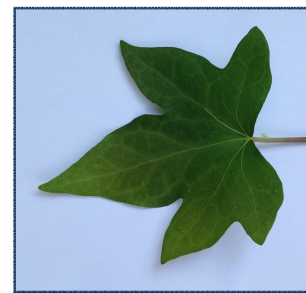


Figura 4.1.14 Hoja Yedra test

Igual que en la anterior imagen las dos hojas están giradas una respecto de la otra, además las dos hojas difieren notablemente la una de la otra en la forma del centro de la hoja (la primera está ligeramente inclinada hacia la derecha)

### Primera imagen

2.50379309060906 4.60895262553261 7.28000718767037 5.90544016222483 12.446442001406 8.20987372890345 12.1612757975165

### Segunda imagen

2.425182442693868 4.522740661863351 6.865274204440954 6.697692594750755 13.365241765083777 8.867211661371657 13.284643964925186

Podemos observar que son valores parecidos. Sin embargo al aplicar la función “Sim” obtenemos el valor **36.5073** sobre los valores de la segunda imagen, que difiere mucho del valor esperado **29**.

Llegados a este punto pueden pasar al menos dos cosas; que los datos enviados a la red no sean los adecuados o que la red este sobre ajustada, teniendo en cuenta que los datos se parecen bastante es muy posible que la red esté sobre ajustada. Para evitar esto se puede intentar entrenar la red con más imágenes del mismo tipo.

En la base de datos había cuatro imágenes de las plantas Higuera, Ficus, Catalpa, Magnolia, Aligustre, Jazmín y Durillo. De ellas tres fueron entrenadas en la red y una fue sometida a test.

En la figura 4.1.15 se muestran los resultados ofrecidos por el grupo de test:

Nombre de Planta	Nº Fotos entrenadas	Nº Fotos de prueba	Acierto
Higuera	4	1	Si
Ficus	4	1	Si
Catalpa	4	1	Si
Magnolia	4	1	Si
Aligustre	4	1	Si
Jazmín	3	1	No
Durillo	2	1	No

Figura 4.1.15

Se puede observar que el porcentaje de aciertos era cercano al 60% (figura 4.1.16)

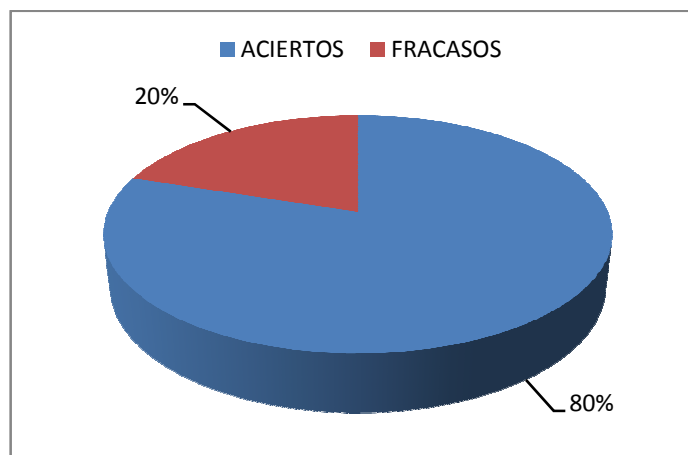


Figura 4.1.16

De los fracasos, la planta Jazmín debía haber dado un valor de **22** y ofreció **23,18**, por lo que se acercó bastante al valor esperado. El “Durillo” quedó a **6,31** de distancia del valor esperado.

De todo el estudio anterior se obtuvieron los parámetros de la red.



### 4.2 Estudio de aplicación Android

El servicio web creado para establecer comunicación con el teléfono Android<sup>23</sup> será el encargado de suministrar las características de la red al terminal que lo solicite. Como la comunicación se realiza vía SOAP, estos datos se pueden consultar en el explorador y se puede comprobar que están en formato XML.

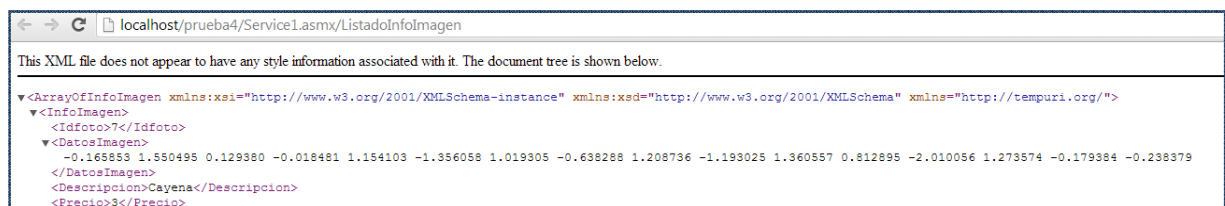


Figura 4.2.1 Servicio Web

El programa del teléfono dispone de un menú que le permite actualizar estos datos del servidor siempre que lo desee.

Además el front-end de la aplicación va buscando contornos y encierra el mayor de ellos en un rectángulo para ayudar a que el usuario sepa cuando el contorno de la hoja está siendo detectado.

Es conveniente hacer la captura sobre un fondo blanco o de un color muy distinto al de la hoja que se pretende identificar además sería deseable tener una buena iluminación. La figura 4.2.2 muestra una imagen capturada en condiciones aceptables en las que la aplicación es capaz de identificar.



Figura 4.2.2

---

<sup>23</sup>Sistema operativo basado en Linux diseñado principalmente para dispositivos móviles.

También se realizaron pruebas para comprobar el grado de fiabilidad del sistema en el teléfono móvil, para ello se tomaron fotos a distintas distancias y con distinta iluminación. A continuación se muestra un ejemplo de resultado obtenido para una hoja de Magnolia.



Figura 4.2.3



Figura 4.2.4



Figura 4.2.5

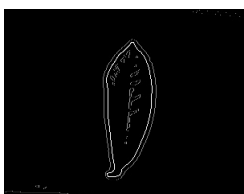


Figura 4.2.6

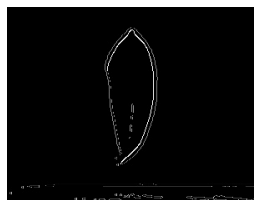


Figura 4.2.7

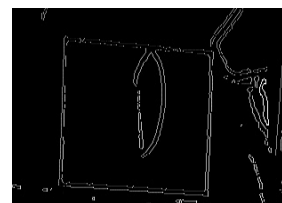


Figura 4.2.8

En la Figura 4.2.3 se puede observar una captura que la aplicación es capaz de identificar debido a que consigue el contorno de una forma bien definida (Figura 4.2.6). Sin embargo, la imagen capturada en la Figura 4.2.4 tiene disminuida significativamente la iluminación respecto de la anterior y el contorno capturado en la Figura 4.2.7 no está bien conseguido, por lo que no es detectable por la aplicación, y la solución aunque se aproxima bastante al valor correcto, no lo consigue. Por último, en la Figura 2.4.5 se muestra una captura de imagen mal realizada, puede observar como la aplicación detecta otros contornos distintos al buscado, vea Figura 2.4.8, ya que en este caso el contorno mayor no es la hoja y por tanto no puede identificarla.

Con estos ejemplos vemos que la aplicación es capaz de funcionar correctamente siempre y cuando la captura de la imagen se haga en unas condiciones adecuadas.

## 5 Conclusiones y trabajo futuro

Teniendo en cuenta la dificultad que entraña la identificación de imágenes basadas en su contenido, los resultados ofrecidos en esta investigación han sido aceptables ya que es capaz de identificar las imágenes que se capturan en buenas condiciones. Además se consiguió demostrar que la infraestructura creada era adecuada para el propósito de la investigación.

Aunque se demostró que el sistema funcionaba, parece obvio que es muy mejorable ya que en muchas ocasiones los momentos Hu no son suficientes para una clasificación adecuada del tipo de hoja. En cualquier caso utilizar los momentos Hu parece un buen comienzo.

Por otro lado se demostró que la infraestructura creada era adecuada, por lo que se alcanzaba uno de los objetivos buscados.

Es una lástima que se dispusiera de una base de datos tan pequeña, puesto que es más difícil conseguir un buen clasificador.

Como trabajo futuro, sería ideal tener una gran base de datos con muchas imágenes de la misma planta y así poder hacer agrupaciones de características de cada tipo de planta para alimentar la red, de esta forma posiblemente se mejorarían los resultados notablemente. Es necesario investigar más en la clasificación e identificación de cada tipo de imagen. Por ejemplo se podrían añadir otro tipo de momentos invariantes como son los de Flusser u otro tipo de características que pudieran ayudar a la clasificación y mejorar los porcentajes de acierto. Sería ideal que la aplicación interactuara más con el usuario, por ejemplo ofreciendo posibles imágenes identificadas. O que el usuario pueda enviar la foto al servidor para poder catalogarla y clasificarla.



## Bibliografía

- [1] Sandra María Palacios. Sistema de reconocimiento de rostros. Universidad peruana de ciencias aplicadas. Archivo electrónico, último acceso el 2 de septiembre de 2013:  
<http://www.cibertec.edu.pe/RepositorioAPS/0/13/JER/PARTICIPACIONENCONGRESOS/Reconocimiento-Rostros.PDF>
- [2] Enrique Cabello Pardos. Técnicas de reconocimiento facial mediante redes neuronales. Tesis Doctoral. Archivo electrónico, último acceso el 2 de septiembre de 2013:  
<http://oa.upm.es/215/1/10200404.pdf>
- [3] Luis Blázquez Pérez. Reconocimiento facial basado en puntos característicos de la cara en entornos no controlados. Archivo electrónico, último acceso el 2 de septiembre de 2013:  
[http://atvs.ii.uam.es/seminars/PFC\\_Luis\\_Blazquez.pdf](http://atvs.ii.uam.es/seminars/PFC_Luis_Blazquez.pdf)
- [4] Juan J. Igarza, Amalur Colau, Iñaki Goirizelaia, Inmaculada Hernáez, Raúl Méndez. Universidad del País Vasco. Sistema de verificación de firmas off-line basado en modelos ocultos de Markov. Archivo electrónico, último acceso el 2 de septiembre de 2013:  
[http://ursi.usc.es/articulos\\_modernos/articulos\\_coruna\\_2003/actas\\_pdf/SESSION%208/S8.%20Aula%202.4/1355-SISTEMA.PDF](http://ursi.usc.es/articulos_modernos/articulos_coruna_2003/actas_pdf/SESSION%208/S8.%20Aula%202.4/1355-SISTEMA.PDF)
- [5] Juan Manuel Pascual Gaspar. Uso de Firma Manuscrita Dinámica para el Reconocimiento Biométrico de Personas en Escenarios Prácticos. Archivo electrónico, último acceso el 2 de septiembre de 2013:  
<http://uvadoc.uva.es/bitstream/10324/130/1/TESIS52-100223.pdf>
- [6] Nancy Rangel Galán, Israel Delgado Guerra, Bravo Zúñiga Haydee. Reconocimiento de caracteres manuscritos. Archivo electrónico, último acceso el 2 de septiembre de 2013:  
<http://www.encuentra.gob.mx/APF?q=RECONOCIMIENTO%20POR%20REGIONES%20DE%20CARACTERES%20MANUSCRITOS&client=cio>
- [7] Leticia María Seijas Reconocimiento de dígitos manuscritos mediante redes neuronales: una técnica híbrida. Archivo electrónico, último acceso el 2 de septiembre de 2013:  
<http://www-2.dc.uba.ar/materias/rn/Aplicaciones/Kohonen/LSeijas-32JAIIO.PDF>
- [8] M. K. Hu, "Visual Pattern Recognition by Moment Invariants", IRE Trans. Info. Theory, vol. IT-8, pp.179–187, 1962

[9] J. Flusser: "On the Independence of Rotation Moment Invariants", Pattern Recognition, vol. 33, pp. 1405–1410, 2000

[10] M. Paz Sesmero Lorente. Diseño análisis y evaluación de conjuntos de clasificadores basados en redes de neuronas. Septiembre-2012. Archivo electrónico, último acceso el 2 de septiembre de 2013:

[http://e-archivo.uc3m.es/bitstream/10016/16177/1/tesis\\_paz\\_sesmero\\_lorente\\_2012.pdf](http://e-archivo.uc3m.es/bitstream/10016/16177/1/tesis_paz_sesmero_lorente_2012.pdf)

[11] Javier Botia, Henry Sarmiento y Claudia Isaza. Redes Neuronales Artificiales de Base Radial como Clasificador Difuso: Una Aplicación en Diagnóstico Médico. Archivo electrónico, último acceso el 2 de septiembre de 2013:

[http://www.academia.edu/1364869/Redes\\_Neuronales\\_Artificiales\\_de\\_Base\\_Radial\\_como\\_Clasificador\\_Difuso\\_Una\\_Aplicacion\\_al\\_Diagnostico\\_Medico](http://www.academia.edu/1364869/Redes_Neuronales_Artificiales_de_Base_Radial_como_Clasificador_Difuso_Una_Aplicacion_al_Diagnostico_Medico)

## [ANEXO A.1]

### RED NEURONAL

Las redes neuronales artificiales son un paradigma de aprendizaje ampliamente tratado en la literatura de la inteligencia artificial<sup>24</sup>. Las redes neuronales artificiales (RNA) intentan imitar la naturaleza de las neuronas del cerebro humano. Uno de los modelos más conocidos son los perceptrones multicapa, cuya estructura se puede apreciar en la figura anexo 1.1<sup>25</sup>.

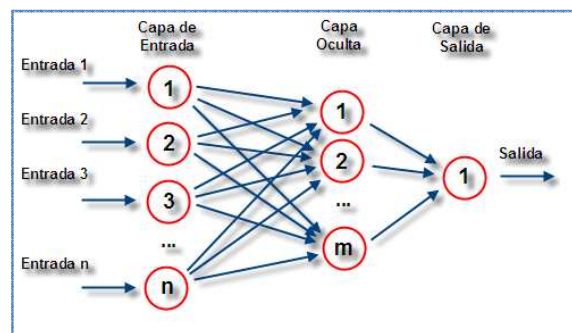


Figura anexo 1.1

Cada neurona tiene una estructura simple. Cada neurona de un vector multicapa recibe el vector de datos de entrada  $\vec{x}$ , y tienen un vector de pesos sinápticos  $\vec{w}$  y una función de activación  $f(x)$ . Dado sus entradas, la salida  $y$  de una neurona es:

$$y = f(\vec{w}, \vec{x})$$

Las redes neuronales artificiales se pueden clasificar según su topología, en las que se distinguen tres tipos básicos de redes:

- Dos tipos de **redes de propagación hacia delante** o acíclicas en las que todas las señales van desde la capa de entrada hacia la salida sin existir ciclos, ni conexiones entre neuronas de la misma capa de red neuronal.
  - Dentro de este grupo podemos encontrar las **Monocapa** y las **Multicapa**, dependiendo del número de capas que tengan.

---

<sup>24</sup> ALPAYDIN, Ethem. Introduction to Machine Learning. Estados Unidos: MIT Press, 2004. 415 p.

<sup>25</sup> Imagen tomada de Wikipedia

- El tercer tipo son las **redes recurrentes** que presentan al menos un ciclo cerrado de activación neuronal.

Una segunda clasificación que se suele hacer es en función del tipo de aprendizaje de que es capaz (si necesita o no un conjunto de entrenamiento supervisado).

- **Aprendizaje supervisado:** necesitan un conjunto de datos de entrada previamente clasificado o cuya respuesta objetivo se conoce. Hay numerosos ejemplos de este tipo de redes, uno de los más clásicos y conocidos son las backpropagation.
- **Aprendizaje no supervisado o autoorganizado:** no necesitan de tal conjunto previo.
- **Redes híbridas:** son un enfoque mixto en el que se utiliza una función de mejora para facilitar la convergencia.
- **Aprendizaje reforzado:** se sitúa a medio camino entre el supervisado y el autoorganizado.

Se pueden destacar como ventajas de las redes neuronales las siguientes:

- **Aprendizaje adaptativo**  
Tienen la Capacidad de aprender a realizar tareas basadas en entrenamiento o en experiencia.
- **Autoorganización**  
La red puede crear su propia organización o su propia representación de la información después de una etapa de aprendizaje.
- **Tolerancia a fallos**  
Una destrucción parcial de la red conduce a una degradación de los resultados pero puede seguir funcionando.
- **Operación en tiempo real**  
Las redes (los computadores neuronales) pueden ser realizadas en paralelo.
- **Fácil inserción dentro de la tecnología existente**  
Una red puede ser rápidamente entrenada, comprobada, verificada y trasladada a hardware con bajo coste, lo que permite su inserción para tareas específicas.



## [ANEXO A.2]

### DISEÑO APLICACIÓN ANDROID

Los pasos para realizar una aplicación para android con eclipse y utilizando las librerías OpenCV vienen detalladamente explicados en <http://opencv.org/platforms/android.html>, por lo que no se mostrarán en esta sección.

A continuación mostraremos el diseño de la aplicación realizada en Android.

En primer lugar usaremos el Diagrama de clases para poder determinar cuál debe ser la mejor estructura de nuestro programa. También usaremos el Diagrama de flujo que nos permite crear las relaciones de dependencia que existen entre los diferentes casos de uso de nuestra aplicación.

Los procesos a realizar son los siguientes:

- Obtener datos del servicio web
- Mostrar imágenes obtenidas por la cámara.
- Captar imagen.
- Obtener los momentos Hu del contorno mayor de la imagen.
- Consultar al clasificador.
- Mostrar resultado.

#### Objetivo

<b>Proyecto:</b>	Proyecto Android
<b>Descripción:</b>	Herramienta computacional que permite obtener datos de imágenes de un servicio web, captar unas imágenes. El sistema procesa las imágenes, extrae sus características e indicar de qué imagen se trata..

## Requerimientos

A continuación se describe una clasificación de los requerimientos de la aplicación Andríd

	Requerimientos
Obtener Datos del servicio web	
R1	Acceder al servicio web y descargar la información en un fichero de texto.
Obtener Imagen y datos imagen	
R2	Capturar la imagen y mostrarla
Procesar imagen	
R3	Obtener el contorno mayor de la imagen y sus características.
Identificar Imagen	
R4	Averiguar de qué imagen se trata.
Guardar imagen	
R5	Guardar imagen, para poder consultarla en otro momento.

## Casos de uso

En este apartado se describen los casos de uso que muestran la funcionalidad completa del sistema, mostrando la interacción con los agentes externos

### Descripción del caso de uso

DESCRIPCION DE CASOS DE USO	
<b>Nombre:</b>	Obtener datos servicio web
<b>Alias:</b>	
<b>Actores:</b>	Usuario
<b>Función:</b>	Obtener datos de imágenes a clasificar del servicio web.
<b>Descripción:</b>	El usuario podrá en cualquier momento descargar la información del tipo de imágenes que desee identificar
<b>Referencias:</b>	De requerimientos: R1

<b>Nombre:</b>	Obtener imagen y sus características
<b>Alias:</b>	
<b>Actores:</b>	Usuario
<b>Función:</b>	Capturar la imagen y conseguir las características.
<b>Descripción:</b>	El sistema debe ser capaz de capturar una imagen y conseguir las

	características del contorno mayor de la imagen capturada.
<b>Referencias:</b>	De requerimientos: R2, R3

<b>Nombre:</b>	Identificar Imagen
<b>Alias:</b>	
<b>Actores:</b>	Usuario
<b>Función:</b>	Identificar de qué imagen se trata.
<b>Descripción:</b>	El sistema, con las características de la imagen junto con la información descargada del servicio web, identificará de qué imagen se trata.
<b>Referencias:</b>	De requerimientos: R4

<b>Nombre:</b>	Guardar datos imagen
<b>Alias:</b>	
<b>Actores:</b>	Usuario
<b>Función:</b>	Guardará la imagen en una carpeta del sistema
<b>Descripción:</b>	Permitirá el almacenamiento de la imagen en una carpeta del sistema para posteriores identificaciones.

<b>Referencias:</b>	De requerimientos: R5
---------------------	-----------------------

### Diagrama de clases

Nos muestra una vista de la aplicación en un determinado momento, es decir, en un instante en que el sistema está detenido. Las clases son la plantilla de los objetos y aquí podemos ver representados a estos con sus atributos o características y su comportamiento o métodos, así como la relación entre ellas.

### Diagrama de Clases:

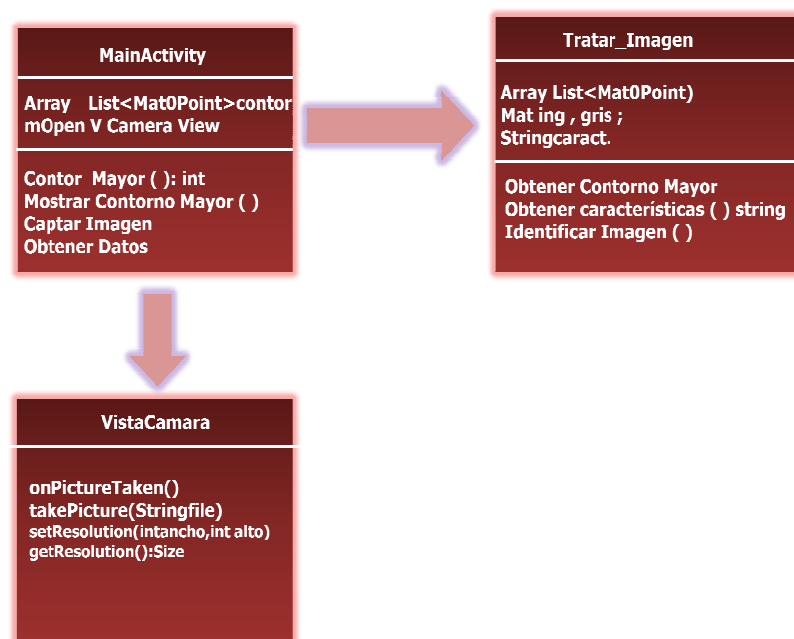


Figura anexo 2.1

### Diagrama de flujo

Permitirá ver los estados por los que pasa nuestra aplicación

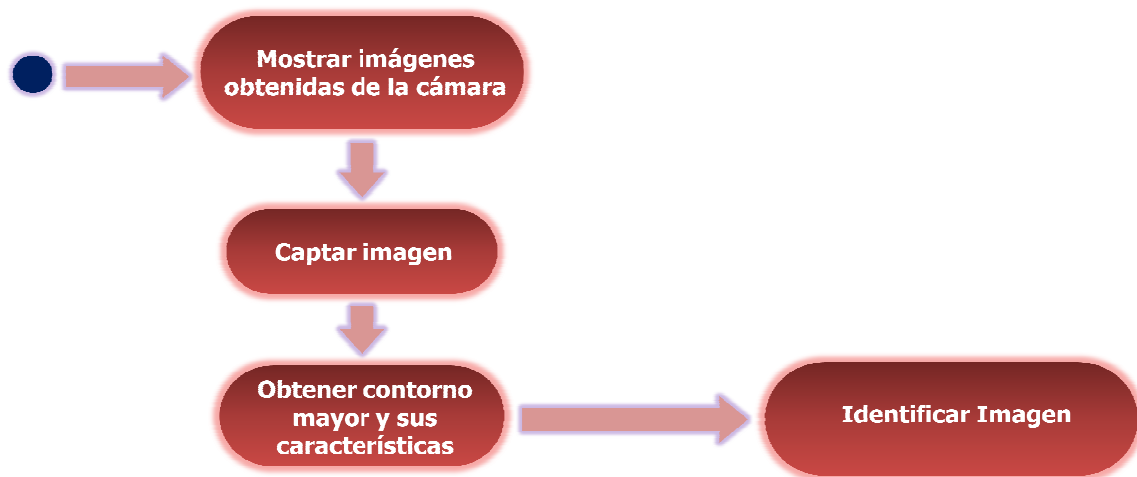


Figura anexo 2.2

## [ANEXO A.3]

### VISUAL STUDIO Y OPENCV

Como se ha comentado, una primera parte consiste en realizar el análisis de las imágenes en el servidor. El programa que extraerá las características de las imágenes será un programa implementado en Visual Studio C++ y utilizará las librerías de OpenCV.

#### Entorno de trabajo

Para el desarrollo de este proyecto se utilizó la plantilla de Visual Studio de “Windows Form”, puesto que apenas hay tutoriales que expliquen cómo hacerlo, se ha decidido incluir unos breves comentarios para ayudar a realizarlo.

Por supuesto un primer paso es instalar las librerías OpenCV, en el sistema operativo, para ello descargamos OpenCV de la página oficial (<http://OpenCV.org/>) y pulsamos ejecutar en el archivo OpenCV.exe. Seleccionamos en la ventana emergente “addOpenCVtothesystempathforalluser”. Se creará una carpeta con todos los “include” de OpenCV.

A continuación se describirán brevemente los pasos realizados para la instalación y configuración de las librerías OpenCV en Visual Studio C++.

- Instalar Visual Studio, se puede descargar una versión “express”, de la página oficial (<http://www.microsoft.com/visualstudio/esn/products/visual-studio-2010-express>).
- Abrimos Visual Studio, seleccionamos “Nuevo Proyecto” y de las plantillas instaladas seleccionamos “vc++” y luego aplicación de “Windows Forms”, asignamos un nombre y pulsamos aceptar.

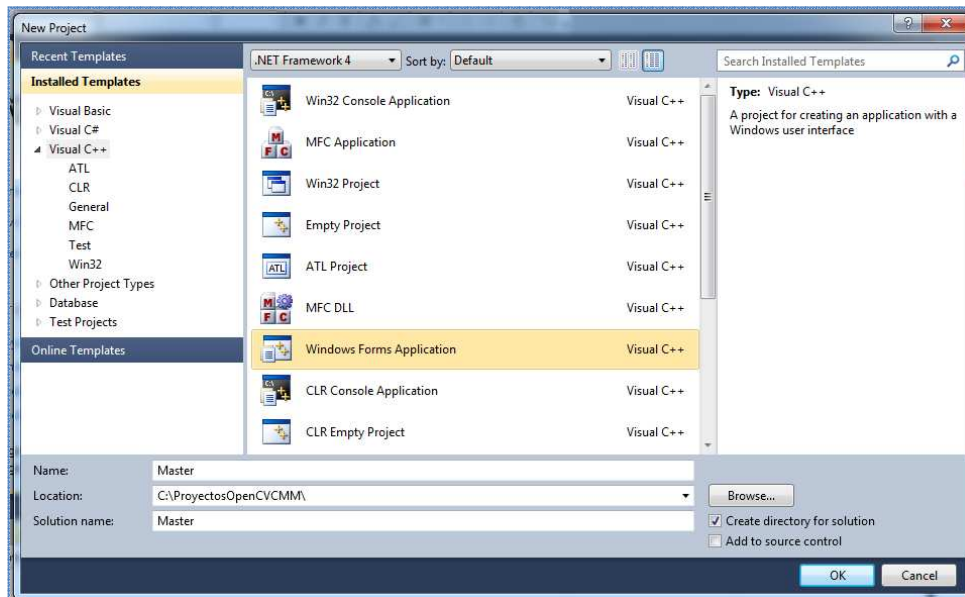


Figura anexo 3.1

- A continuación empezamos a configurar, para ello, en el menú de Visual Studio, pulsamos en “proyecto/propiedades”, en el menú de la izquierda elegimos “propiedades de configuración/general” y en el menú de la derecha en “Common Language Runtime Support”, elegimos “Common Language Runtime Support (/clr)”. El problema de elegir esta opción es que durante el desarrollo del programa, no podremos hacer uso del Intellisense” del IDE de Visual Studio, lo cual dificulta notablemente el desarrollo del mismo.



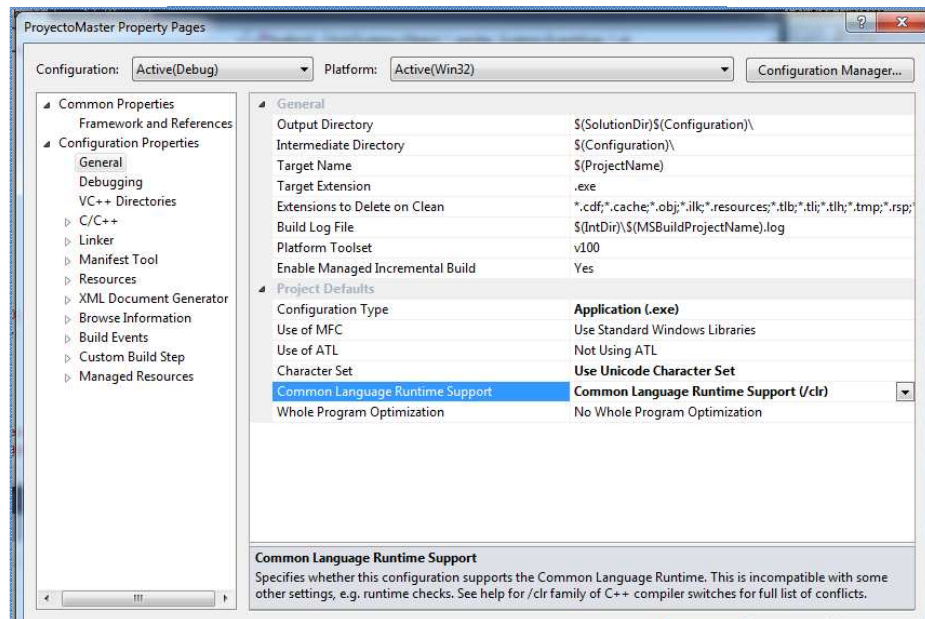


Figura anexo 3.2

- Pulsamos en “VC++ Directories” y en “Includedirectories”, se añade lo siguiente “C:\OpenCV\build\include” junto con “C:\ProgramFiles\Microsoft SDKs\Windows\v7.0A\Include\gl”

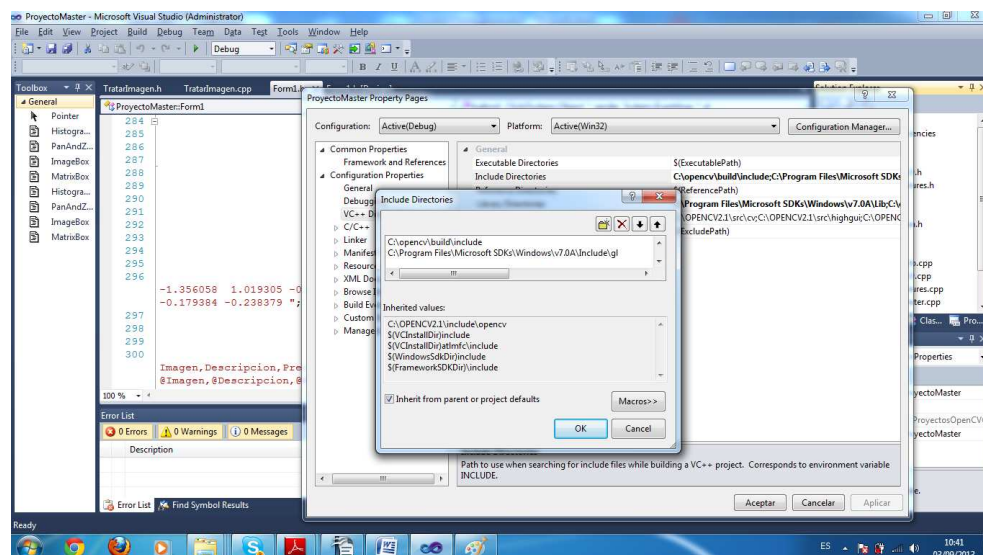


Figura anexo 3.3

- En Library Directories añadimos “C:\OpenCV\build\x86\vc10\lib” y “C:\Program Files\Microsoft SDKs\Windows\v7.0A\Lib”.
- Por último en “Linker/Input”, en el cuadro de la izquierda en “Additional Dependencies” elegimos las librerías que vayamos a necesitar, si no se sabe, se pueden añadir todas.

### Diseño

Aunque no realizaremos un diseño formal, basándonos en todas las reglas de la metodología UML<sup>26</sup> utilizaremos dos diagramas que nos han servido para determinar el esqueleto principal de nuestra aplicación. Y también, nos permitirán tener una visión gráfica de lo que estamos construyendo.

### Objetivo

<b>Proyecto:</b>	Proyecto fin de Máster
<b>Descripción:</b>	Herramienta computacional que permite acceder a unas imágenes y a un fichero de texto almacenadas en una carpeta. El sistema procesa las imágenes, extrae sus características y las guarda en un fichero. También inserta información relacionada con las imágenes junto con estas en una base de datos Microsoft Sql Server.

### Requerimientos

A continuación se describe una clasificación de los requerimientos del proyecto

---

<sup>26</sup>Lenguaje de Modelado Unificado, es una especificación de notación orientada a objetos

	Requerimientos
Obtener Imagen y datos imagen	
R1	Acceder a la imagen y leer información asociada a ella
Procesar imagen	
R2	Extraer características imagen.
R3	Generar fichero con características imagen procesada
Características Red Neuronal	
R4	Llamar programa Matlab y extraer características red neuronal
Guardar imagen	
R5	Guardar imagen, junto con características Red neuronal devuelto por programa Matlab y datos imagen en Base de Datos.

### Casos de uso

En este punto se mostrarán los Casos de uso que describen la funcionalidad completa del sistema, mostrando la interacción con los agentes externos

### Descripción del caso de uso

DESCRIPCION DE CASOS DE USO	
<b>Nombre:</b>	Obtener datos imagen
<b>Alias:</b>	
<b>Actores:</b>	Administrador
<b>Función:</b>	Obtener las imágenes junto con los datos para procesarlas.
<b>Descripción:</b>	El Administrador registrará las imágenes en una carpeta junto con un fichero descriptivo de cada imagen. El sistema las leerá y preparará para su proceso.
<b>Referencias:</b>	De requerimientos: R1

<b>Nombre:</b>	Procesar Imagen
<b>Alias:</b>	
<b>Actores:</b>	Administrador
<b>Función:</b>	Procesar las imágenes para extraer características y llamar a función de Matlab.
<b>Descripción:</b>	El sistema una vez lea la imagen, las procesará y extraerá las características de las mismas y las guardará en un fichero
<b>Referencias:</b>	De requerimientos: R2, R3

<b>Nombre:</b>	Características Red Neuronal
<b>Alias:</b>	
<b>Actores:</b>	Administrador
<b>Función:</b>	Llamar a programa externo de Matlab para generar red neuronal y extraer datos red neuronal
<b>Descripción:</b>	El sistema llamará a programa de Matlab y extraerá las características de la red neuronal generada por Matlab.
<b>Referencias:</b>	De requerimientos: R4

<b>Nombre:</b>	Guardar datos imagen
<b>Alias:</b>	
<b>Actores:</b>	Administrador
<b>Función:</b>	Insertar en Base de datos las imágenes, información asociada y características red neuronal
<b>Descripción:</b>	El sistema llamará a programa de Matlab y extraerá las características de la red neuronal generada por Matlab.
<b>Referencias:</b>	De requerimientos: R5

### Diagrama de clases

Nos muestra una vista de la aplicación en un determinado momento, es decir, en un instante en que el sistema está detenido. Las clases son la plantilla de los objetos y aquí podemos ver representados a estos con sus atributos o características y su comportamiento o métodos, así como la relación entre ellas.

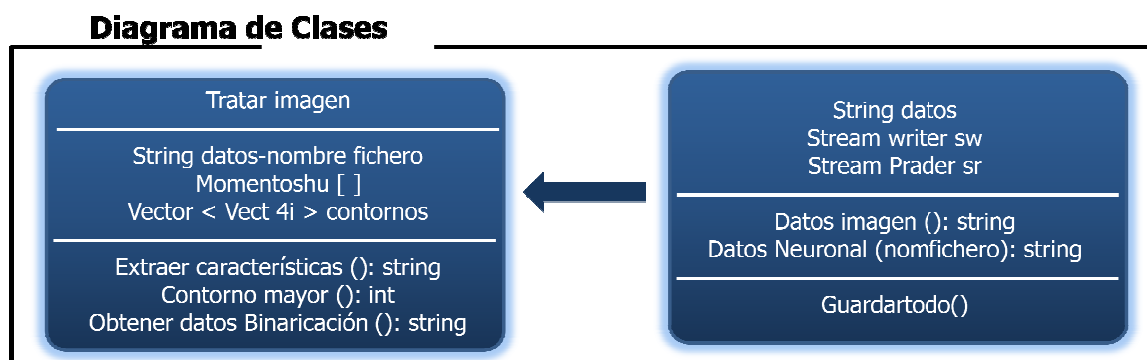


Figura anexo 3.4

### Diagrama de flujo

Permitirá ver los estados por los que pasa nuestra aplicación

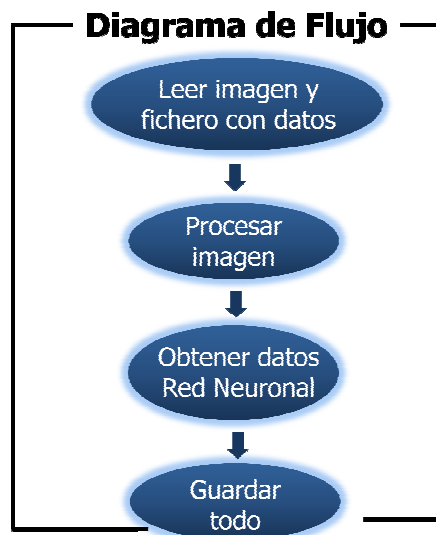


Figura anexo 3.5

## ANEXO A.4

Se ha investigado otro tipo de imágenes para ayudar a evaluar el sistema diseñado. La idea de este estudio consistió en proponer otro tipo de características representativas de imágenes distintas a los momentos Hu. En este caso se analizaron imágenes de zapatillas deportivas. Se abordó el problema desde varios puntos de vista.

Por un lado se intentó el estudio de un promedio local de bordes para reducir el número de puntos a estudiar.



Figura anexo 4.1



Figura anexo 4.2

El método consistía en convertir la imagen a escala de grises y posteriormente se binarizaba la imagen y se obtenían los contornos de la imagen. Posteriormente se analizaban los contornos por regiones y allí donde había un porcentaje adecuado de contorno, esa región se ponía a 1, el resto a 0. Con esto se reducía el número de puntos que era necesario para alimentar la red neuronal y obtener el clasificador. Sin embargo aunque pudiera funcionar para imágenes ideales, en imágenes obtenidas con el teléfono, se demostró que no era viable este método.

Se realizó otro estudio, intentando obtener gamas de colores azul, rojo o verde. En las figuras 4.3, 4.4 y 4.5 se muestra la descomposición en colores de la figura 4.1. La idea consistía en separar la imagen en tres planos por separado e intentar detectar contornos y características en cada uno de estos colores. También se hacía un thresholding de la imagen en escala de grises. El método se realizó de la siguiente forma:

1. Se realizaba un thresholding de 128 como umbral de separación para quedarse con lo más rojo, verde o azul.

2. Detectar contornos en la imagen.
3. Con los contornos detectados hay varias alternativas
  - Basarse en los puntos del contorno; Nos quedamos con un número fijo de puntos (por ejemplo 20) del contorno<sup>27</sup>
  - Para cada plano de imagen con sus 20 puntos se entrenaba un clasificador

El resultado de la investigación anterior falló porque OpenCV puede decidir empezar a dar un contorno por el lado derecho y el siguiente por el izquierdo. Por ejemplo si en un contorno toda la suela se aproxima por una sola línea mientras que otro aproxima por tres.



Figura anexo 4.3



Figura anexo 4.4



Figura anexo 4.5

Este estudio demostró que la clasificación de este tipo de imágenes es más complicada y necesita de una mayor investigación, sin embargo permitió comprobar que la infraestructura diseñada era la adecuada para el estudio de cualquier tipo de imagen..

---

<sup>27</sup> <http://www.codeproject.com/Articles/114797/Polyline-Simplification>)